Bedienungsanleitung

Programmierbarer Einschub des Messverstärkersystems MG*Cplus*

ML70B

A0862-3.0 de

Inhalt **Seite**

Sicherheitshinweise **5**

1 Einführung **7**

2 Frontplatte **8**

3 Anschlussplatten **9**
3.1 Anschlussplatte AP71 11
3.2 Anschlussplatte AP72 11
3.3 Anschlussplatte AP75 12
3.4 Anschlussplatte AP78 13

4 Arbeitsweise des ML70B **14**
4.1 Zeitverhalten 14
4.2 Messwertübertragung 15
4.3 Programmstruktur 16
4.4 Programme speichern 16
4.5 Das Hauptprogramm PLC_PRG 18

5 Einführung in das Programmieren **19**
5.1 Programmiersystem installieren 19
5.2 Programmbeispiel 20
5.3 Projekt übersetzen 26
5.4 Zielsystem starten und Programm laden 27

6 Kommunikation mit anderen Verstärkerkanälen **27**
6.1 Messwerte einlesen 27
6.2 Befehle senden 29

7 Hardwarekomponenten konfigurieren **29**
7.1 Analogausgänge 29
7.2 Leuchtdioden der Frontplatte 30
7.3 Berechnete Werte ausgeben 30

8 Ansteuern der Anschlussplatten **31**
8.1 AP71/CAN 31
8.1.1 Grundsätzliche Funktion des CAN-Treibers 31
8.2 Serielle Kommunikation (AP72) 33
8.2.1 Daten senden 34
8.2.2 Daten empfangen 34

8.3 Digitale Ein- und Ausgänge (AP75) 36
8.4 Analogausgänge AP78 37

9 Dialoge erzeugen 37
9.1 Dialogstruktur 37
9.2 Parametereigenschaften 38
9.2.1 Eigenschaften des Parametertyps NODE 39
9.2.2 Eigenschaften der Parametertypen DINTPAR, INTPAR, REALPAR 40
9.2.3 Eigenschaften des Parametertyps KEY 40
9.2.4 Eigenschaften des Parametertyps TEXT 41
9.2.5 Eigenschaften des Parametertyps MENUE 41

10 Einstellparameter speichern 46

11 Mehrprogrammbetrieb 48

12 Sonderfälle 49
12.1 Äquidistante Messwertausgabe 49
12.2 Anzahl der Unterkanäle ändern 49

13 Debugfunktionen (PLC−Browser) 50

14 Systemvariablen 51

15 Fehlermeldungen 52

16 Technische Daten ML70B 53

17 Technische Daten Anschlussplatten 54
17.1 AP71 54
17.2 AP72 54
17.3 AP75, AP78 55

Sicherheitshinweise

Bestimmungsgemäße Verwendung

Der programmierbare Einschub ML70B ist ausschließlich für Messaufgaben und direkt damit verbundene Steuerungsaufgaben zu verwenden. Jeder darüber hinausgehende Gebrauch gilt als nicht bestimmungsgemäß.

Zur Gewährleistung eines sicheren Betriebes darf das Gerät nur nach den Angaben in der Bedienungsanleitung betrieben werden. Bei der Verwendung sind zusätzlich die für den jeweiligen Anwendungsfall erforderlichen Rechts- und Sicherheitsvorschriften zu beachten. Sinngemäß gilt dies auch bei Verwendung von Zubehör.

Allgemeine Gefahren bei Nichtbeachten der Sicherheitshinweise

Der Einschub ML70B entspricht dem Stand der Technik und ist betriebssicher. Von dem Gerät können Restgefahren ausgehen, wenn es von ungeschultem Personal unsachgemäß eingesetzt und bedient wird.

Jede Person, die mit Aufstellung, Inbetriebnahme, Wartung oder Reparatur des Gerätes beauftragt ist, muss die Bedienungsanleitung und insbesondere die sicherheitstechnischen Hinweise gelesen und verstanden haben.

Restgefahren

Der Leistungs- und Lieferumfang des ML70B deckt nur einen Teilbereich der Messtechnik ab. Sicherheitstechnische Belange der Messtechnik sind zusätzlich vom Anlagenplaner/Ausrüster/Betreiber so zu planen, zu realisieren und zu verantworten, dass Restgefahren minimiert werden. Jeweils existierende Vorschriften sind zu beachten. Auf Restgefahren im Zusammenhang mit der Messtechnik ist hinzuweisen.
Sollten Restgefahren beim Arbeiten mit dem ML70B auftreten, wird in dieser Anleitung mit folgenden Symbolen darauf hingewiesen:

Symbol: **WARNUNG**

Bedeutung: **Möglicherweise gefährliche Situation**

Weist auf eine **mögliche** gefährliche Situation hin, die − wenn die Sicherheitsbestimmungen nicht beachtet werden − Tod oder schwere Körperverletzung zur Folge **haben kann**.

Sicherheitsbewusstes Arbeiten

Fehlermeldungen dürfen nur quittiert werden, wenn die Ursache des Fehlers beseitigt ist und keine Gefahr mehr existiert.

Das Gerät entspricht den Sicherheitsanforderungen der DIN EN 61010-Teil1 (VDE 0411-Teil1); Schutzklasse I.

Um eine ausreichende Störfestigkeit zu gewährleisten, nur die *Greenline*-Schirmführung verwenden (siehe HBM-Sonderdruck "*Greenline*-Schirmungskonzept, EMV-gerechte Messkabel; G36.35.0).

Umbauten und Veränderungen

Der Einschub ML70B darf ohne unsere ausdrückliche Zustimmung weder konstruktiv noch sicherheitstechnisch verändert werden. Jede Veränderung schließt eine Haftung unsererseits für daraus resultierende Schäden aus.

Insbesondere sind jegliche Reparaturen, Lötarbeiten an den Platinen untersagt. Bei Austausch gesamter Baugruppen sind nur Originalteile von HBM zu verwenden.

Qualifiziertes Personal

Dieses Gerät ist nur von qualifiziertem Personal ausschließlich entsprechend der technischen Daten in Zusammenhang mit den nachstehend aufgeführten Sicherheitsbestimmungen und Vorschriften einzusetzen bzw. zu verwenden. Bei der Verwendung sind zusätzlich die für den jeweiligen Anwendungsfall erforderlichen Rechts- und Sicherheitsvorschriften zu beachten. Sinngemäß gilt dies auch bei Verwendung von Zubehör.

Qualifiziertes Personal sind Personen, die mit Aufstellung, Montage, Inbetriebsetzung und Betrieb des Produktes vertraut sind und die über die ihrer Tätigkeit entsprechende Qualifikationen verfügen.

Wartungs- und Reparaturarbeiten am geöffneten Gerät unter Spannung dürfen nur von einer ausgebildeten Person durchgeführt werden, die sich der vorliegenden Gefahr bewusst ist.

1 Einführung

Der programmierbare Einschub ML70B ist ein 4 Teileinheiten breites Modul, das einen Steckplatz im MGC*plus*-Systemgerät belegt. Der Einschub ist mit dem Programmiersystem “CoDeSys” nach dem international genormten SPS-Programmierstandard IEC61131−3 frei programmierbar.

Was kann der ML70B?

- Messwerte von beliebigen anderen MGC*plus*-Einschüben mit einer Abtastrate von 2400 Hz verarbeiten, auch Daten, die über CAN- oder Profibus in das Systemgerät eingelesen werden.
- Befehle an andere Verstärker schicken.
- berechnete Werte als Messwerte im MGC*plus* ausgeben, die vom Anzeige- und Bedienfeld AB22A und vom MGCplus-Assistenten angezeigt und weiterverarbeitet werden können.
- bis zu zwei Anschlussplatten der Typen AP71, AP72, AP75, AP78 ansteuern (Kombinationsmöglichkeiten siehe Seite 9).

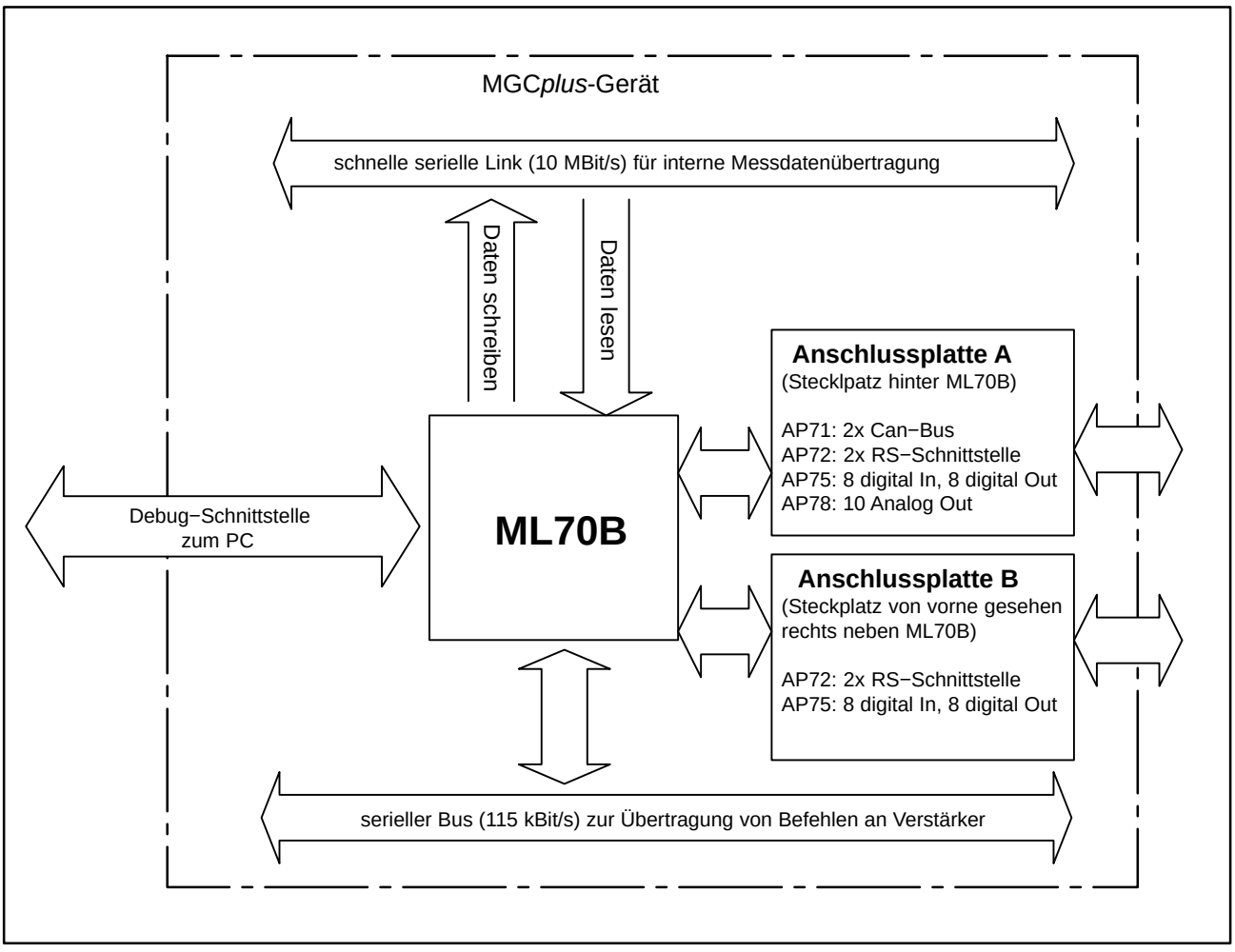
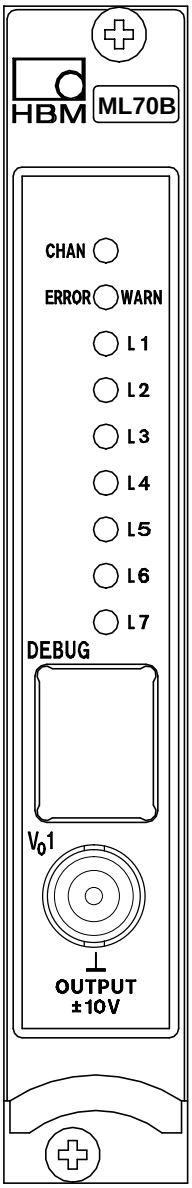


Abb.1.1: ML70B im MGC*plus*-Gerät

2 Frontplatte



Beschriftung	Farbe	Bedeutung
CHAN.	Gelb	Kanal selektiert
ERROR/WARN	Rot	Fehler
L1	Rot, Gelb	Nach Programmierung
L2	Rot, Gelb	Nach Programmierung
L3	Rot, Gelb	Nach Programmierung
L4	Rot, Gelb	Nach Programmierung
L5	Rot, Gelb	Nach Programmierung
L6	Rot, Gelb	Nach Programmierung
L7	Rot, Gelb	Nach Programmierung

Die LEDs L1...L7 sind programmierbar (siehe Seite 30).

3 Anschlussplatten

Der ML70B kann bis zu zwei der folgenden Anschlussplatten ansteuern:
AP71 (2x CAN−Bus)
AP72 (2x RS-232/RS-485/RS-422; einzeln per Software umschaltbar)
AP75 (8x Digital−In, 8x Digital−Out)
AP78 (8x Analog−Out)

Jede der aufgeführten Anschlussplatten kann einzeln angesteuert werden. Folgende Kombinationen sind möglich:

AP71, AP72
AP71, AP75
AP72, AP72
AP72, AP75
AP75, AP72
AP75, AP75
AP78, AP72
AP78, AP75

Die Anschlussplatten können direkt hinter dem Verstärkereinschub oder rechts daneben (bei Ansicht von oben) eingesteckt werden (siehe Abb.3.1).

HINWEIS

Nur die Anschlussplatte hinter dem Einschub ML70B hat die analogen Ausgangsspannungen V_{O1} und V_{O2}!

Die Ausgänge der Anschlussplatte AP78 sind in den Einstellmenüs mit **AO** (Analogue Output) gekennzeichnet.

Befindet sich die Anschlussplatte AP75 direkt hinter dem Einschub, sind die Ein- und Ausgänge mit **A.** gekennzeichnet. Befindet sie sich rechts daneben, wird den Eingangs-/Ausgangsnummern **B.** vorangestellt.

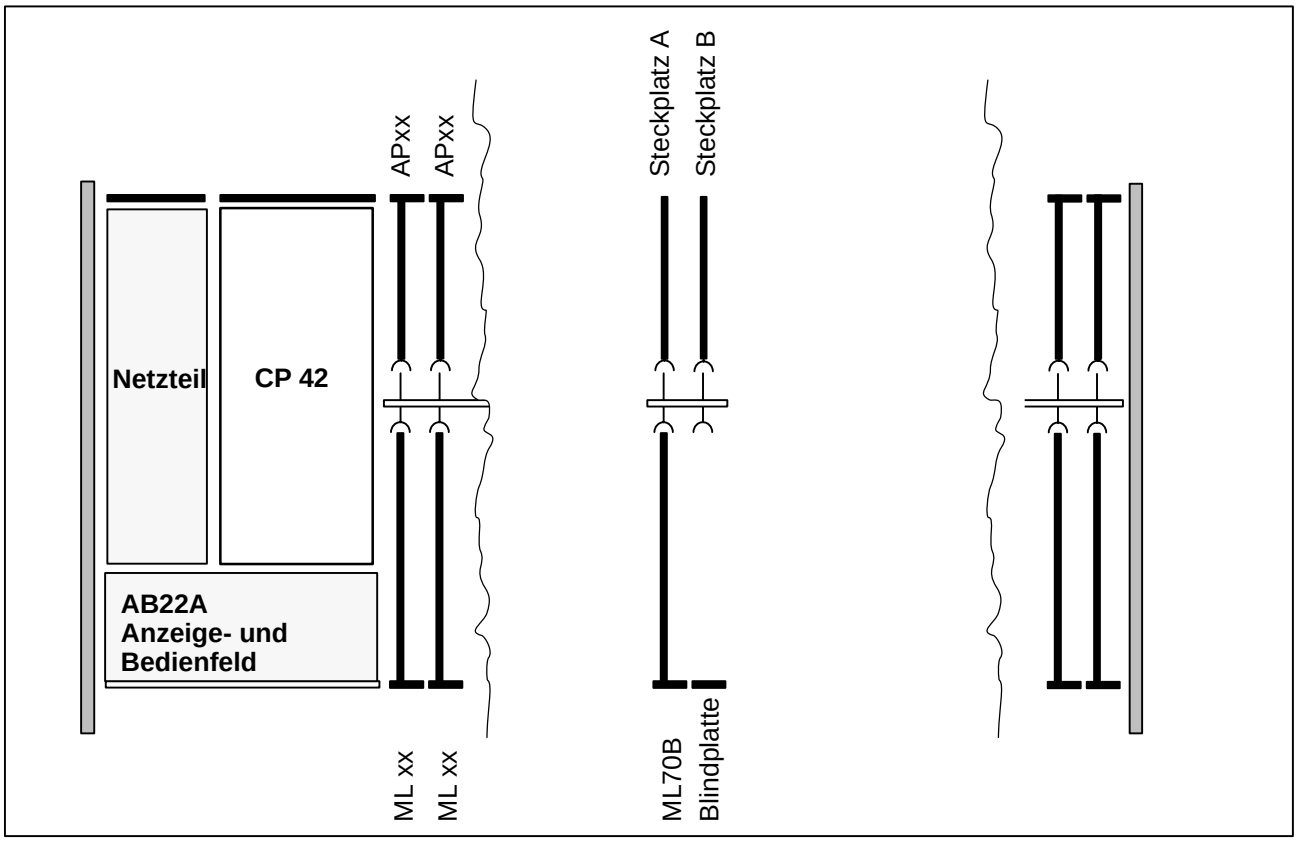
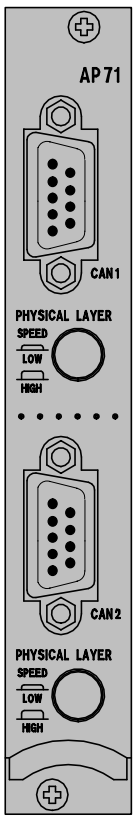
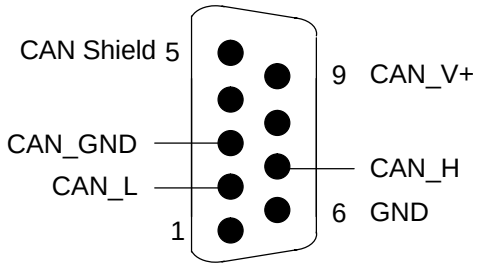


Abb.3.1: Anschlussplatten-Kombinationen im Systemgerät (Ansicht von oben)

3.1 Anschlussplatte AP71



Anschlussstecker
CAN1/CAN2



Physical Layer:

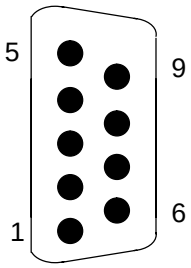
High: Standard CAN-Geschwindigkeiten bis 1MBaud nach ISO11898 (24V Spannungsfestigkeit).

Low: Komfortbus im Fahrzeug bis 100kBaud (CAN_V+ muß an 12V angeschlossen sein)

Abb.3.2: Steckerbelegung AP71

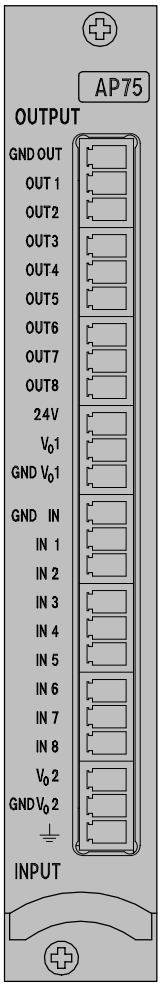
3.2 Anschlussplatte AP72

Anschlussstecker



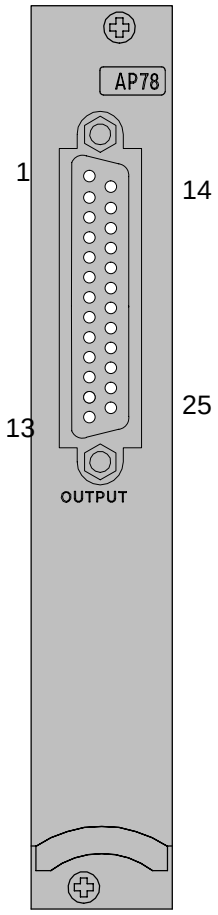
Pin	Funktion RS232	FunktionRS422	FunktionRS485
1	CD (nicht verschaltet)	–	–
2	RXD – Receive Data	RXA–Receive Data A (nicht invertierend)	–
3	TXD – Transmit Data	TXB – Transmit Data B (invertierend)	RXB–Receive Data B (invertierend)
4	DTR –12 V über Vorwiderstand	–	–
5	GND	GND	GND
6	DSR (nicht verschaltet)	–	–
7	RTS – Request To Send	TXA–Transmit Data A (nicht invertierend)	RXA–Receive Data A (nicht invertierend)
8	CTS – Clear To Send	RXB–Receive Data B (invertierend)	–
9	RI (nicht verschaltet)	–	–

3.3 Anschlussplatte AP75



Die Anschlussplatte AP75 hat acht digitale Eingänge und acht digitale Ausgänge. Die Ein- und Ausgänge sind einzeln galvanisch getrennt und haben eigene Massesysteme (GND OUT: Masse für Ausgänge; GND IN: Masse für Eingänge).

3.4 Anschlussplatte AP78



Pin	Funktion
1	Analogausgang Anschlussplatte AO3
2	Analogausgang Anschlussplatte AO4
3	Analogausgang Anschlussplatte AO5
4	Analogausgang Anschlussplatte AO6
5	Analogausgang Anschlussplatte AO7
6	Analogausgang Anschlussplatte AO8
7	Analogausgang Anschlussplatte AO9
8	Analogausgang Anschlussplatte AO10
9	–
10	–
11	Analogausgang Verstärker VO1
12	Analogausgang Verstärker VO2
13	–
14	GND zu AO3
15	GND zu AO4
16	GND zu AO5
17	GND zu AO6
18	GND zu AO7
19	GND zu AO8
20	GND zu AO9
21	GND zu AO10
22	–
23	–
24	GND zu VO1
25	GND zu VO2

4 Arbeitsweise des ML70B

4.1 Zeitverhalten

Der ML70B hat folgende Aufgaben:

- Einlesen der Messwerte
- Ausführen des geladenen Programmes
- Bedienung der Anschlussplatten
- Kommunikation mit den MGC*plus*-Komponenten (CP42, Verstärkerkanäle, ext. Rechner usw.)

Diese Aufgaben werden zyklisch abgearbeitet (siehe Abb.4.1). Die Kommunikation läuft dabei dauernd im Hintergrund.

Die Messwerte werden interruptgesteuert mit einer Frequenz von 2400 Hz eingelesen. Mit dieser Rate fallen auch die Messdaten intern im MGC*plus* an. Nach dem Einlesen der Messwerte werden die Anschlussplatten bedient und das Anwenderprogramm wird ausgeführt.

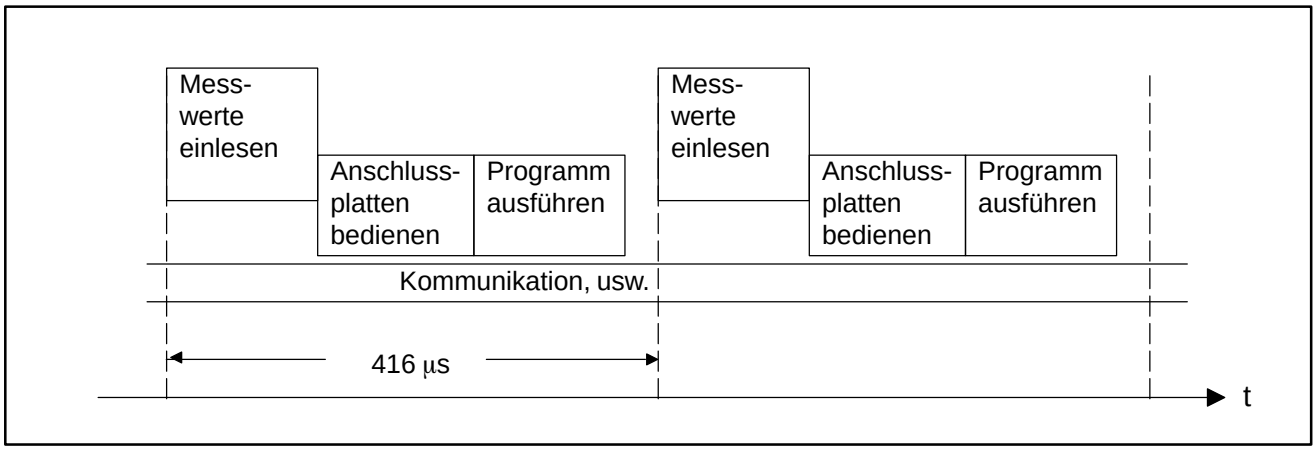


Abb.4.1: Zykluszeit des ML70B

Die maximale Aufruffrequenz von 2400 Hz wird nur dann erreicht, wenn das IEC-Programm in weniger als 400 µs (1/2400 Hz) wieder verlassen wird. Dauert ein Schritt länger, verringert sich die Aufruffrequenz entsprechend (nächstkleinere Frequenz 1200 Hz).

Deshalb sollte jedes Programm so formuliert werden, dass es in jeder Situation möglichst schnell verlassen wird. Diese Eigenschaft unterscheidet SPS−Programme grundlegend von “normalen” Programmen.

4.2 Messwertübertragung

Verstärkereinschübe (ML ...) und Kommunikationsprozessor (CP ...) im MGC*plus* sind mit einer schnellen synchronen Datenschnittstelle (Link) verbunden, über die Messwerte im System ausgetauscht werden. Die Daten werden mit einer Frequenz von 2400 Hz ausgetauscht.

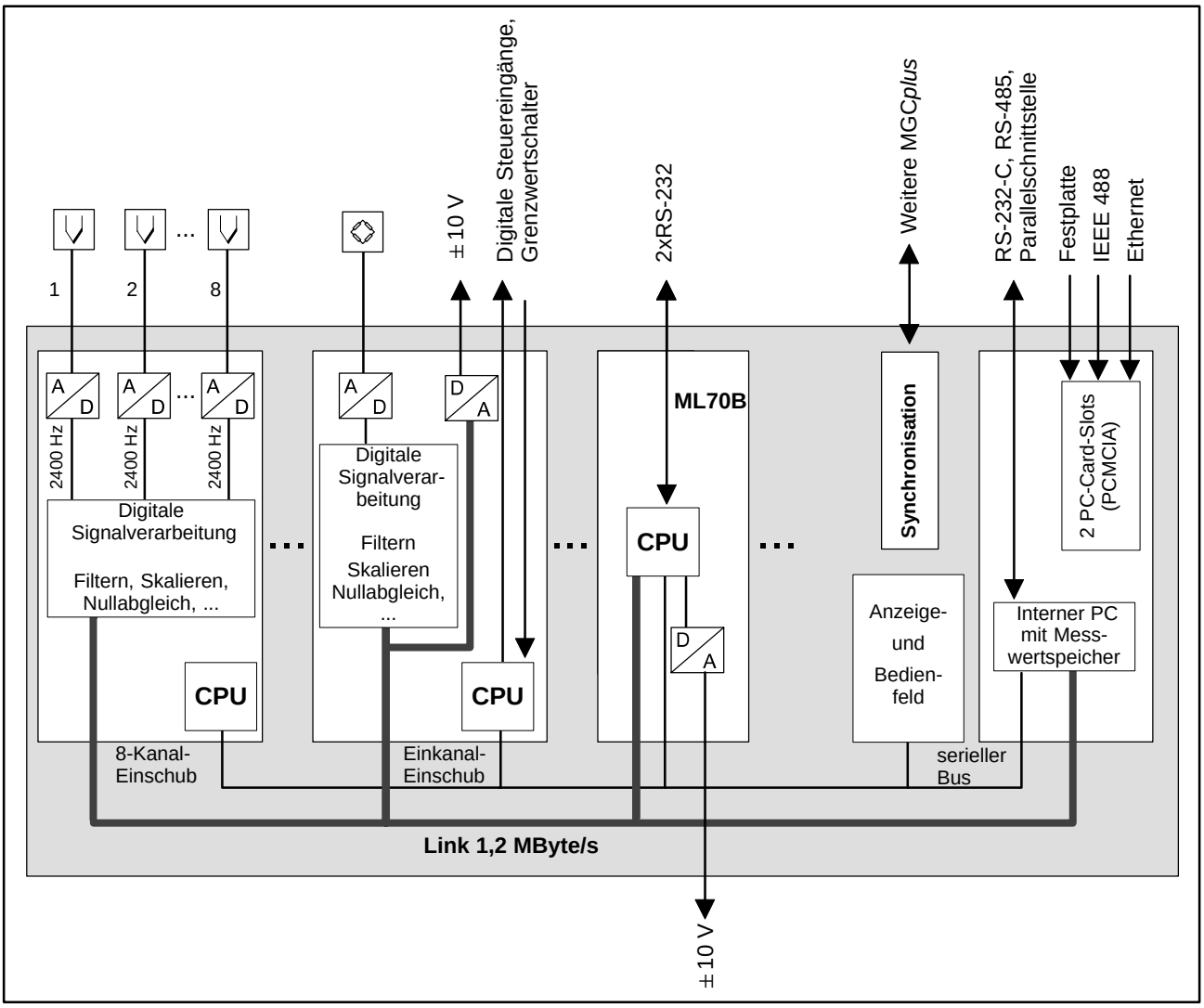


Abb.4.2: Kommunikation innerhalb des MGCplus

Jeder Einschub kann insgesamt 8 Messwerte mit einer Übertragungsrate von je 2400 Hz ausgeben. Das bedeutet für Einkanalverstärker, dass mehrere Signale (Brutto, Netto, Spitzenwert, ...) gleichzeitig ausgeben werden können.

Mehrkanalverstärker können für jeden der acht Unterkanäle immer nur ein Signal pro Unterkanal ausgeben, da insgesamt nur acht übertragen werden können. Es kann bei einem Mehrkanalverstärker z. B. nicht gleichzeitig ein Brutto− und ein Nettomesswert von einem Unterkanal angefordert werden.

Werden von mehreren Komponenten im MGCplus Messwerte angefordert, muss die Bedienreihenfolge festgelegt werden. Der Kommunikationsprozessor (CP42) verwaltet den Datentransport, bei ihm müssen benötigte Messwerte angefordert werden. Jede Komponente, die Messwerte benötigt, muss diese beim Kommunikationsprozessor anmelden.

Zu Konflikten kann es kommen, wenn z. B. von einem 8-Kanal-Verstärker zwei Signale gleichzeitig angefordert werden. In diesem Fall wird die zweite Anforderung mit einer Fehlermeldung quittiert.

Grundsätzlich gilt beim Anmelden von Link-Ressourcen: solange Ressourcen verfügbar sind, werden diese vergeben und zwar in der Reihenfolge der Anforderung.

4.3 Programmstruktur

Ein ML70B-Programm besteht aus den im vorherigem Kapitel erläuterten Gründen immer aus folgenden Schritten:

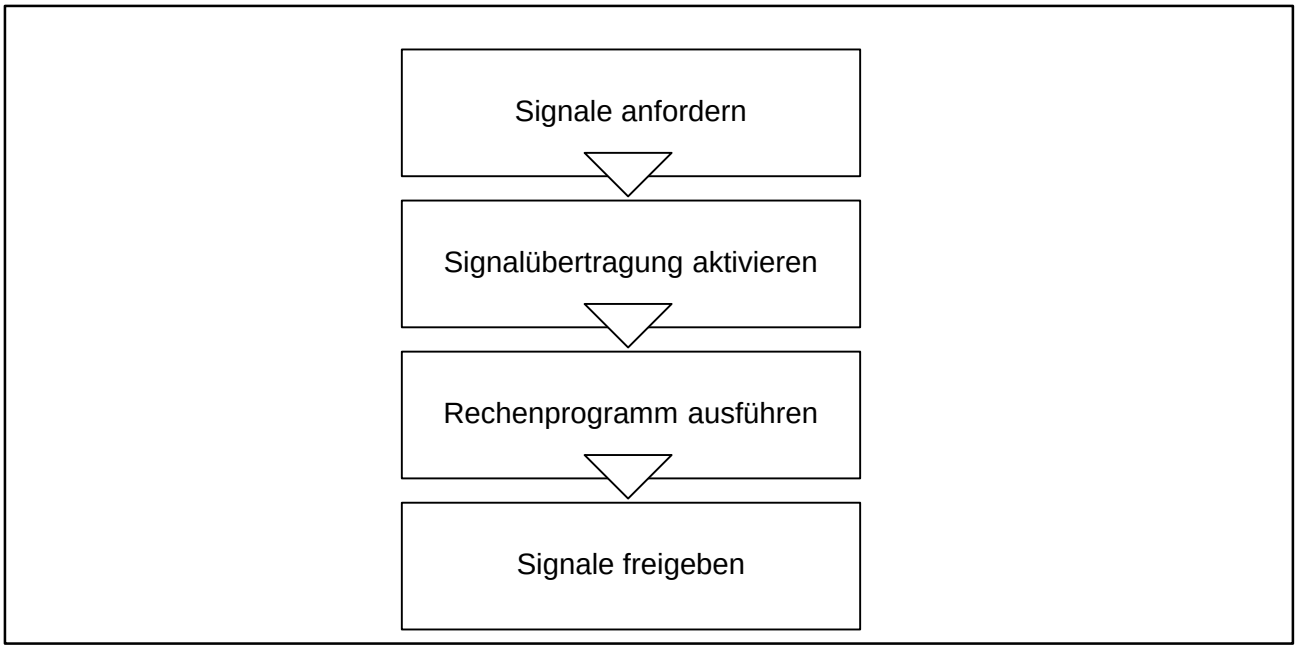


Abb.4.3: Programmstruktur

Daraus ergibt sich für alle Projekte, bei denen Messwerte vom MGC*plus* verarbeitet werden sollen, die gleiche Programmstruktur. Wir empfehlen, alle Messprogramme wie im Kapitel 4.5 beschrieben aufzubauen.

4.4 Programme speichern

Ein Programm wird im ML70B im internen flüchtigen Speicher (RAM) ausgeführt.

Beim Einschalten des MGC*plus* wird geprüft, ob im nichtflüchtigen Speicher (FLASH) ein Programm vorhanden ist. Wenn ja, wird das gespeicherte Programm aus dem FLASH in den flüchtigen Speicher kopiert und automatisch ausgeführt.

Vom Programmiersystem kann ein anderes Programm in den ML70B geladen und getestet werden, ohne das vorhandene Programm im dauerhaften Speicher zu verändern. Solange der RAM-Inhalt nicht ins FLASH kopiert wird,

führt der ML70B nach Aus− und Wiedereinschalten des MGC*plus* das alte Programm aus dem FLASH aus (siehe auch Kap.10).

Den Flash-Inhalt können Sie bei Bedarf löschen:

1. Schalten Sie das MGCplus-Gerät aus.
2. Ziehen Sie den ML70B-Einschub aus dem Gehäuse. Auf der Unterseite des Einschubes befindet sich eine 10polige Steckerleiste.
3. Stecken Sie eine Brücke auf Pin 9 und Pin10.
4. Schalten Sie das MGCplus-Gerät wieder ein, der Flash-Inhalt wird gelöscht.
5. Schalten Sie das MGCplus-Gerät wieder aus und entfernen Sie die Brücke.

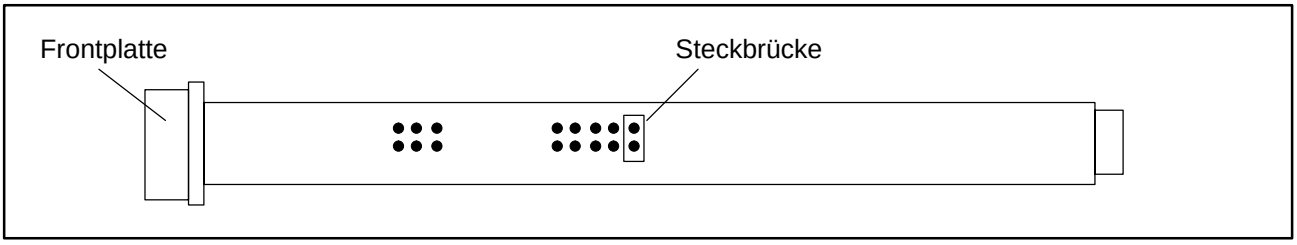


Abb.4.4: Verstärkereinschub ML70B; Ansicht von unten

Soll ein neues Programm dauerhaft im ML70B gespeichert werden, wird es aus dem RAM in das FLASH-ROM kopiert (siehe CoDeSys “Bootprojekt erzeugen”).

4.5 Das Hauptprogramm PLC_PRG

Das Hauptprogramm PLC_PRG wird vom ML70B synchron zu den Messwerten aufgerufen. Jedes Programm sollte nach folgendem Schema in der Programmiersprache AS (Ablaufsprache) aufgebaut sein:

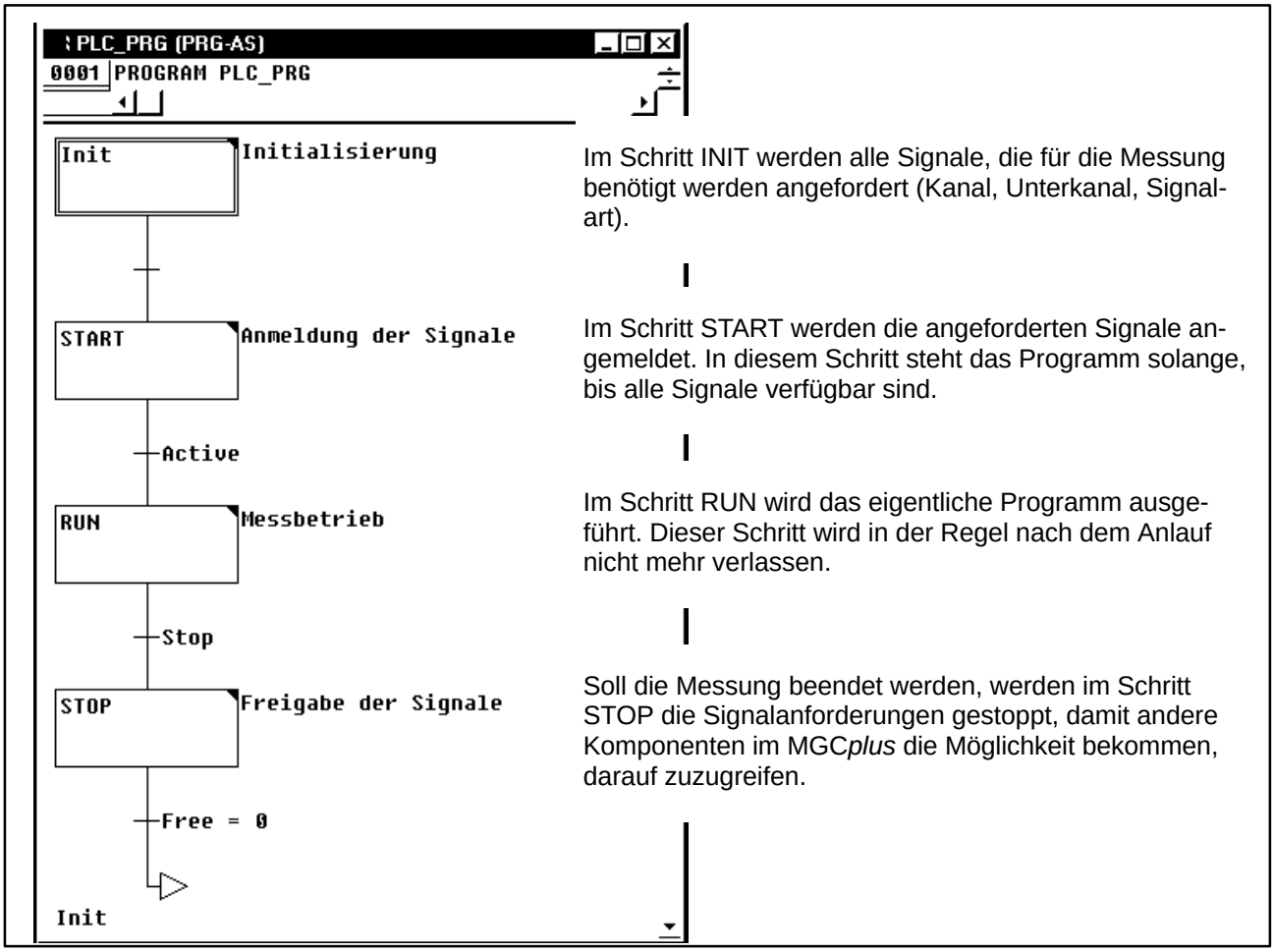

PLC_PRG (PRG-AS)
0001 PROGRAM PLC_PRG
Init
Initialisierung
Im Schritt INIT werden alle Signale, die für die Messung benötigt werden angefordert (Kanal, Unterkanal, Signalart).
START
Anmeldung der Signale
Im Schritt START werden die angeforderten Signale angemeldet. In diesem Schritt steht das Programm solange, bis alle Signale verfügbar sind.
Active
RUN
Messbetrieb
Im Schritt RUN wird das eigentliche Programm ausgeführt. Dieser Schritt wird in der Regel nach dem Anlauf nicht mehr verlassen.
Stop
STOP
Freigabe der Signale
Soll die Messung beendet werden, werden im Schritt STOP die Signalanforderungen gestoppt, damit andere Komponenten im MGC*plus* die Möglichkeit bekommen, darauf zuzugreifen.
Free = 0
Init

Abb.4.5: Programmablauf im ML70B

Mit diesem Schema wird lediglich die Anforderung und Freigabe von Messwerten festgelegt. Die eigentliche Funktion des Programmes beinhalten die einzelnen Schritte.

Der Inhalt der einzelnen Schritte kann in jeder beliebigen IEC61131−3−Sprache formuliert sein und wird mit einem Doppelklick auf den einzelnen Schritt geöffnet.

HINWEIS

Das dargestellte Schema ist nicht wie ein “normales” Flussdiagramm zu lesen! Bei jedem Aufruf des Programmes wird nur ein einzelner Schritt bearbeitet und das Programm anschließend wieder verlassen. Ist die zwischen zwei Schritten stehende Übergangsbedingung (Transition) erfüllt, wird beim nächsten Programmaufruf der nachfolgende Schritt bearbeitet. Ist die Bedingung nicht erfüllt, wird beim nächsten Aufruf noch einmal der gleiche Schritt ausgeführt.

5 Einführung in das Programmieren

In diesem Kapitel wird anhand eines einfachen Programmbeispieles das Programmiersystem CoDeSys und die Bedienung des ML70B erläutert. Eine detaillierte Beschreibung des Programmiersystems entnehmen Sie bitte der Online−Dokumentation der CoDeSys-CD.

Systemvorraussetzungen:

- MGC*plus*-Gerät mit:

  AB22A

  CP42 (optional)

  Kanal1: ML55B, ML01B o.ä. (Einkanalverstärker)

  Kanal2: ML55B, ML01B o.ä. (Einkanalverstärker)

  Kanal3: ML70B mit/ohne Anschlussplatte
- PC mit Windows-Betriebssystem und serieller Schnittstelle

5.1 Programmiersystem installieren

1. Legen Sie die CoDeSys−CD in das CD−ROM−Laufwerk Ihres Rechners. Das Setup startet automatisch; sollte dies nicht der Fall sein, starten Sie die Datei **Setup.exe** auf der CD mit Doppelklick manuell.
2. Wählen Sie die gewünschte Setup-Sprache und klicken Sie auf ***OK.***
3. Schließen Sie das Fenster “Willkommen...” durch Klicken auf ***Weiter.***
4. Wählen Sie Im Fenster Setuptypauswahl ***Entwicklungssystem*** und klicken Sie auf ***Weiter*.**
5. Im Fenster Komponentenauswahl sollten alle Checkboxen aktiviert sein (Default-Einstellung). Behalten Sie die Einstellung des Zielordners bei und klicken Sie auf ***Weiter.***

6. Wählen Sie die gewünschte Sprache der Programmoberfläche und klicken Sie auf ***Weiter.***
7. Wählen Sie den gewünschten Ordner aus und klicken Sie auf ***Weiter.***

Nach der Anzeige der Zusammenfassung erfolgt die Installation des Entwicklungssystems.

Bei der mitgelieferten Installations-CD wird das Zielsystem ML70B (Target) automatisch installiert. Sie können das Zielsystem aber auch nachträglich installieren:

1. Starten Sie über ***Start*** → ***Programme*** → ***CoDeSys V2.2*** das Programm ***InstallTarget***.
2. Öffnen Sie mit der Schaltfläche ***Öffnen..*** die Datei <CD−Laufwerk>/Hottinger/install/Hottinger.tnf.
3. Markieren Sie den Eintrag “Hottinger Baldwin Messtechnik GmbH” im linken Fenster und betätigen Sie die Schaltfläche ***Installieren*****.** Beantworten Sie die Frage “Das Installationsverzeichnis existiert nicht...?” mit Ja.

Im rechten Fenster **Installierte Zielsysteme** sollte jetzt der Eintrag “Hottinger Baldwin Messtechnik GmbH” vorhanden sein. Damit ist die Installation abgeschlossen.

5.2 Programmbeispiel

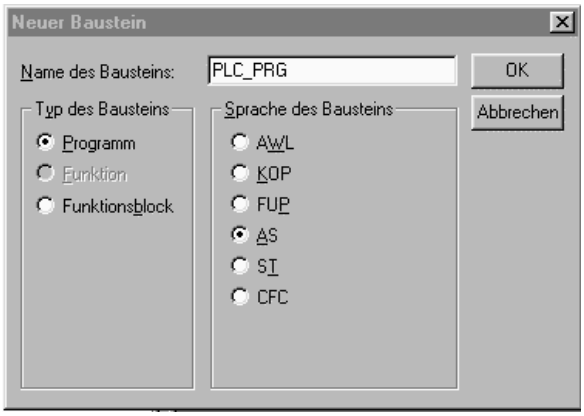
Zur Bestimmung der Leistung eines Motors soll das gemessene Drehmoment mit der gemessenen Drehzahl multipliziert werden. Das Ergebnis soll in der Anzeige des MGC*plus*-Gerätes bzw. im Assistenten angezeigt werden.

Formel zur Berechnung der Leistung:

$$Leistung\,[W] = Drehmoment\,[N \cdot m] \cdot Drehzahl\left[\frac{1}{\mathrm{min}}\right] \cdot \frac{2 \cdot \pi}{60}$$

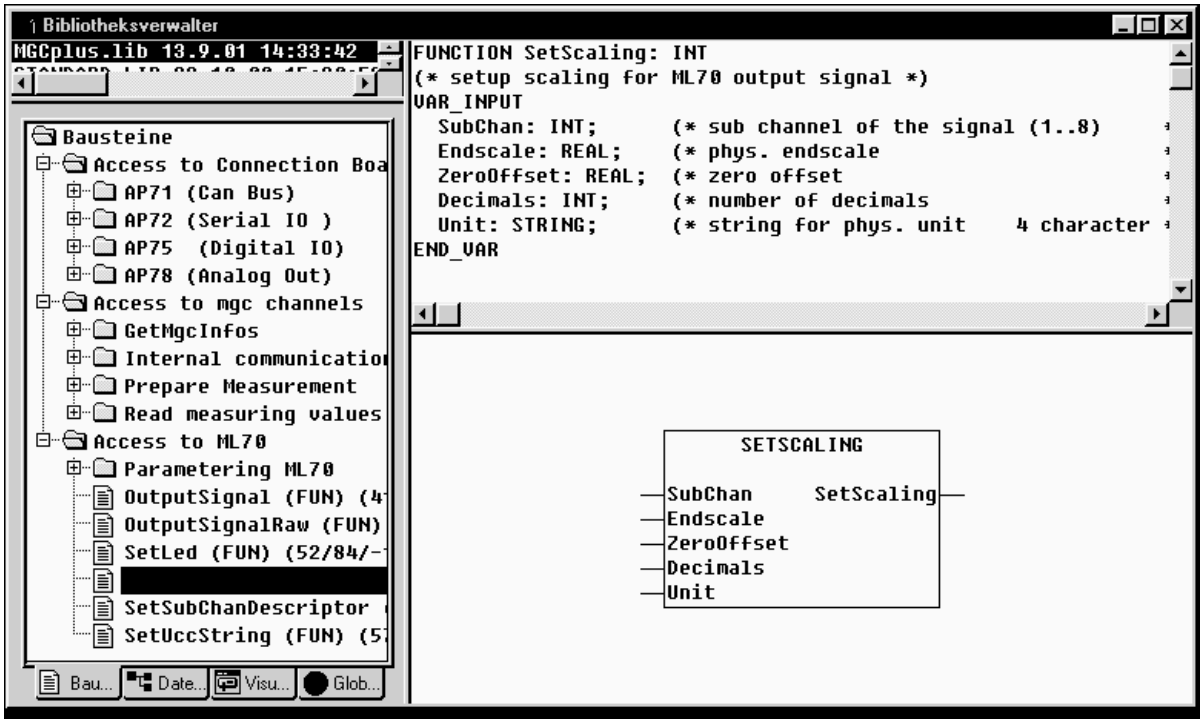
1. Starten Sie das Programmiersystem CoDeSys über ***Start*** → ***Programme*** → ***CoDeSys V2.2.***
2. Erzeugen Sie über ***Datei*** → ***Neu*** ein neues Projekt.
3. Wählen Sie das Zielsystem *HBM_ML70B* aus und bestätigen Sie mit ***OK*** (danach wird automatisch das Fenster *Neuer Baustein* geöffnet).

4. Wählen Sie den Baustein *PLC_PRG* vom Typ *Programm* in der Sprache *AS* (Ablaufsprache) aus und bestätigen Sie mit ***OK.***

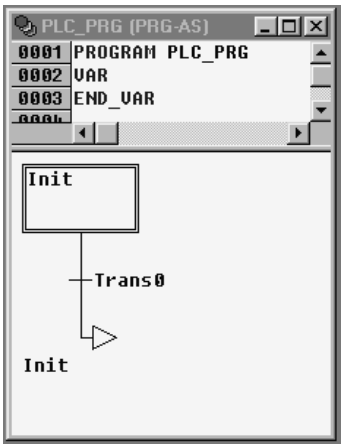


5. Um aus dem Programm auf MGC*plus*-Komponenten zugreifen zu können, müssen Sie die Bibliothek MGCPLUS.LIB laden. Öffnen Sie ***Fenster→Bibliotheksverwaltung*** und fügen Sie mit ***Einfügen→Weitere Bibliothek →MGCplus.lib*** die Bibliothek ein.

Die Bibliothek beinhaltet sämtliche Funktionen, um auf Hardwarekomponenten des MGC*plus* zugreifen zu können. Jede verfügbare Funktion ist als Baustein mit kurzen Erläuterungen über die Bedeutung der Parameter dargestellt. Spezielle Rechenvorschriften usw. sind in eigenen Bibliotheken verpackt.



Nachdem Sie das neue Projekt angelegt haben, sehen Sie im Arbeitsbereich folgendes Fenster:



Programmstruktur des Hauptprogrammes erstellen

Das Hauptprogramm soll, wie in Kapitel 4.5 beschrieben, aus vier Schritten bestehen.

6. Klicken Sie auf den Trennstrich der Transition ***Trans0*** so, dass ein gepunkteter Rahmen um Trans0 erscheint und wählen Sie im Kontextmenü (rechte Maustaste klicken) den Befehl ***Schritt−Transition (danach)***. Wiederholen Sie diesen Vorgang zweimal bis Schritt 4 (Step 4).
7. Wenn Sie direkt auf den Namen einer Transition oder eines Schrittes klicken, wird dieser blau markiert und kann geändert werden. Benennen Sie auf diese Weise die drei neuen Schritte in **START**, **RUN** und **STOP** um.

Kommentare eingeben

8. Markieren Sie einen Schritt durch Klicken knapp außerhalb des Rahmens, so dass ein gepunkteter Rahmen um den Schritt erscheint. Klicken Sie mit der rechten Maustaste auf den Schritt und wählen Sie im Kontextmenü ***Schrittattribute*** aus. In dem danach geöffneten Fenster können Sie Ihre Kommentare eintragen.
9. Damit die Kommentare rechts neben den Schritten angezeigt werden, wählen Sie im Menü ***Extras −> Optionen...*** die Option ***Kommentar*** aus.

Schritt “Init” programmieren

Die benötigten Signale im System werden über die Funktion *RequestSignal()* aus der Bibliotheksverwaltung aufgerufen. Als Aufrufparameter müssen Kanalnummer, Unterkanalnummer, Signalnummer übergeben werden. Die Codierung der Signaltypen ist im Fenster *“Bibliotheksverwalter”* erläutert. Als Rückgabewert liefert die Funktion eine Kennung (Handle), über die später der gewünschte Messwert abgefragt werden kann.

In unserem Beispiel wird das Drehmomentsignal von Kanal 1, Unterkanal 1 als Bruttosignal gemessen und die Drehzahl als Nettosignal von Kanal 2, Unterkanal 1.

10. Doppelklicken Sie auf den Schritt “Init”. Sie werden nach der gewünschten Sprache gefragt. Wählen Sie **ST** (Strukturierter Text) und es erscheint ein neues Fenster für die Aktion zu diesem Schritt. Geben Sie in die ersten zwei Programmzeilen ein:

    HandleTorque := RequestSignal(1,1,0);

    HandleRpm := RequestSignal(2,1,1);

Die Variablen `“HandleTorque”` und `“HandleRpm”` müssen Sie als Integerzahl (INT) deklarieren.

Danach müssen Sie die Skalierung für das Ergebnis der Rechnung festlegen:

Maximaler Wert des Drehmomentes ist z. B. 500 N·m, maximale Drehzahl 10000 min^{-1}.

Die maximal zu messende Leistung beträgt damit:

$$10000 \cdot 100 \cdot 2 \cdot 3{,}14/60\ W = 104666\ W$$

11. Geben Sie in die dritte Programmzeile ein:
    SetScaling(1,120000.0,0.0, 1,’W’);

Die Parameter bedeuten:

1	Skalierung für Unterkanal 1 des ML70B
120000.0	Maximalwert, der bei der Rechnung auftreten kann
0.0	Keine Nullverschiebung
1	Das Rechenergebnis soll mit einer Nachkommastelle angezeigt werden
’W’	Die Einheit des Messwertes ist ’W’ (Watt)

Damit ist die Programmierung des Schritts “Init” abgeschlossen und Sie können das Fenster **Aktion Init** schließen. In der linken oberen Ecke des Schrittes INIT erscheint nun ein kleines Dreieck, welches signalisiert, dass dieser Schritt programmiert ist.

Schritt “START” programmieren

Die erforderlichen Signale werden durch den Aufruf der Funktion *ActivateSignals()* im System angefordert. Der Schritt START wird solange aufgerufen, bis die Freigabe der Messsignale erfolgt ist. Dann ist die Bedingung zum Übergang in den Schritt RUN erfüllt.

Geben Sie für den Schritt START folgende Programmzeile ein:

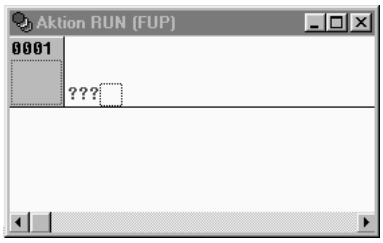
Ready := ActivateSignals(SIGNAL_REQUEST);

Deklarieren Sie die Variable Ready als INT.

Schritt “START” programmieren

Der Schritt START wird in der Sprache FUP (Funktionsplan) programmiert, hier findet die eigentliche Berechnung statt. Dieser Schritt wird in unserem Programmbeispiel nicht mehr verlassen.

1. Doppelklicken Sie auf den Schritt RUN und wählen Sie die Sprache FUP. Es erscheint folgendes Fenster:

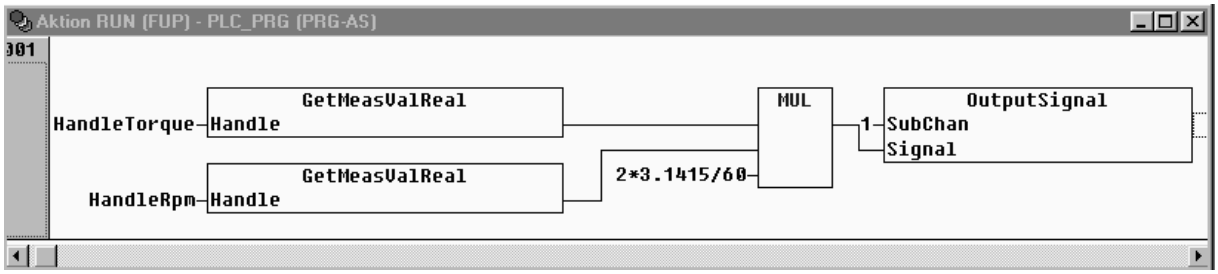


2. Setzen Sie den Mauszeiger in die quadratische Markierung hinter den drei Fragezeichen und führen Sie über das Kontextmenü (rechte Maustaste) den Befehl ***Baustein*** aus. Es wird standardmäßig der Baustein ***AND*** eingefügt, der Name ist markiert.
3. Drücken Sie die Taste F2 (Eingabehilfe). Sie erhalten einen Dialog, in dem Sie aus den verfügbaren Bausteinen auswählen können. Wählen Sie für unser Beispiel zunächst im linken Fenster die Kategorie ***Standard-Funktionen*** aus. Wählen Sie dann im rechten Fenster aus der Bibliothek **Targets\Hottinger\...\MGCplus.lib** im Ordner **Access to ML70B** die Funktion **OutputSignal** aus.
4. Klicken Sie jetzt die drei Fragezeichen neben der Eingangsvariable **SubChan** an, und geben Sie den Wert ‚1‘ ein.
5. Markieren Sie den Anschlussstrich neben der Eingangsvariablen **Signal** so, dass er durch einen gepunkteten Rahmen markiert ist und führen Sie im Kontextmenü den Befehl **Baustein** aus.
6. Ersetzen Sie den markierten Text ‚AND‘ durch Eingabe von ‚MUL‘ (für Multiplikation).
7. Markieren Sie wieder den Anschlussstrich neben der oberen Eingangsvariable von ‚MUL‘ und führen Sie über das Kontextmenü den Befehl **Baustein** aus.
8. Drücken Sie F2 und fügen Sie die Funktion **Standard-Funktionen → MGCplus.lib → Read measuring values → GetMeasValReal**.
9. Wiederholen Sie den Vorgang für die zweite Eingangsvariable von ‚MUL‘. Geben Sie als Eingangsvariablen von GetMeasValReal() die Variablen

HandleTorque und HandleRpm ein. Jetzt müssen Sie noch den Korrekturfaktor für die Multiplikation einfügen.

10. Klicken Sie dazu auf den Baustein **MUL** und führen Sie über das Kontextmenü den Befehl **Eingang** aus. Es erscheint ein weiterer Eingang mit ??? als Eingangsgröße. Markieren Sie die drei Fragezeichen und ersetzen Sie sie durch den Ausdruck **2 * 3.1415/60**

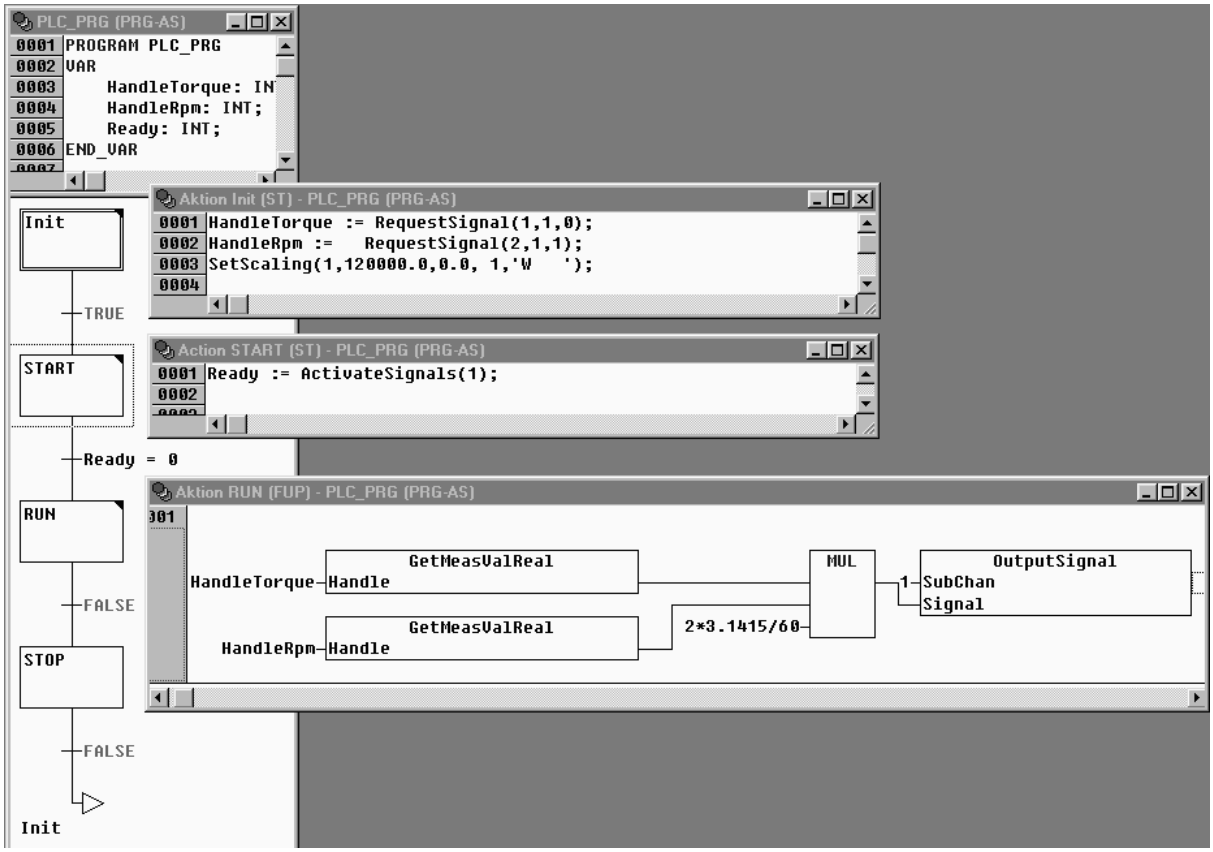
Damit ist die Programmierung des Schrittes RUN abgeschlossen. Das entstandene Netzwerk sollte jetzt folgendermaßen aussehen:



Transitionen festlegen

Transitionen sind Übergangsbedingungen von einem Schritt in den nächsten. Überschreiben Sie die erste Transition nach Init mit dem Wert TRUE . Der Schritt Init wird damit genau einmal aufgerufen. Die zweite Transition nach START erhält die Bedingung Ready = 0 (die im Schritt Start gesetzte Variable). Wenn Ready = TRUE (d.h. wenn die Anmeldung der Signale abgeschlossen ist) beginnt der nächste Schritt.
Die dritte und vierte Transition erhalten den Wert FALSE, d.h. der Schritt RUN wird nie mehr verlassen.

Die Programmierung ist nun abgeschlossen. Das entstandene Programm sollte jetzt folgendermaßen aussehen:



5.3 Projekt übersetzen

Übersetzen Sie das Projekt mit dem Menübefehl ***Projekt → Alles übersetzen*** oder drücken Sie die Funktionstaste **<F11>.** Im Meldungsfenster rechts unten sollte nach dem Übersetzungslauf **“0 Fehler”** gemeldet werden. Wenn nicht, überprüfen Sie bitte die Richtigkeit aller Eingaben. Beachten Sie dazu auch die Fehlermeldungen.

Durch Drücken der Funktionstaste **<F4>** werden Sie direkt zu den Fehlern im Programm geführt.

5.4 Zielsystem starten und Programm laden

Das Zielsystem für dieses Beispiel ist ein MGC*plus* mit folgender Bestückung:

Anzeige- und Bedienfeld AB22A
Kommunikationsprozessor CP42 (optional)
Kanal 1: ML55B, ML01B o. ä. (Einkanalverstärker)
Kanal 2: ML55B, ML01B o. ä. (Einkanalverstärker)
Kanal 3: ML70B

Der PC und ML70B kommunizieren über eine serielle Verbindung. Verbinden Sie den ML70B über die DEBUG-Buchse auf der Frontplatte durch das mitgelieferte Kabel mit einer seriellen Schnittstelle des PCs. Um die Schnittstelle einzurichten gehen Sie wie folgt vor:

- Führen Sie den Menübefehl **Online → Kommunikationsparameter** aus. Wählen Sie in dem erscheinenden Dialog die Schaltfläche ***Neu***, um den Verbindungsaufbau zu Ihrem Zielsystem zu konfigurieren.
- Vergeben Sie in dem neuen Dialog einen geeigneten Namen für die Verbindung und selektieren Sie ***Serial RS232***. Durch mehrfaches Klicken auf die Schnittstelle COM1 können Sie auf COM2,3,.. umschalten. Verlassen und beenden Sie den Dialog mit OK.
- Stellen Sie mit dem Menübefehl ***Online → Einloggen*** im Programmiersystem **CoDeSys** die Verbindung zum Zielsystem her und starten Sie mit ***Online → Start*** Ihr Programm auf dem Zielsystem.

Am AB22A ist jetzt das Ergebnis der Multiplikation ablesbar, wenn der ML70B über die Kanalwahltaste CHANNEL + / − angewählt ist. Am Bildschirm des PCs können Sie jetzt in den einzelnen Fenstern die Variablen beobachten.

Soll das Programm dauerhaft im ML70B gespeichert werden, müssen Sie das Menü ***Online → Bootprojekt erzeugen*** aufrufen. Damit wird das Programm in den nichtflüchtigen FLASH−Speicher übertragen. Beim nächsten Einschalten des MGC*plus* wird dieses Programm dann automatisch gestartet.

6 Kommunikation mit anderen Verstärkerkanälen

6.1 Messwerte einlesen

Vor dem Einlesen der Messwerte aus anderen MGCplus-Verstärkerkanälen müssen zunächst die benötigten Signale angefordert und aktiviert werden (siehe Kap.4.5ff.)

Die Messwerte werden über die Funktionen `GetMeasValReal()` und `GetMeasValRaw()` eingelesen.

Beispiel:

Messwert: REAL

Handle: INT

MessWert := GetMeasValReal(Handle)

Die Variable “Handle” muss vorher im Schritt START zugewiesen worden sein (siehe Kap.4.5 und Kap. 5.2)

Auf die Messwerte wird dann nur noch über dieses Handle zugegriffen. Die Funktion `GetMeasValReal()` liefert dann immer den Momentanwert für das jeweilige Handle als physikalisch skalierte Größe.

GetMeasValRaw() liefert einen unskalierten Rohwert als 32−Bit−Integerzahl im Format DINT. Dieses Datenformat ist dann vorzuziehen, wenn aus Zeitgründen auf Rechnungen im REAL−Format verzichtet werden muss. Der Rohwert wird dabei nach folgender Formel in den physikalischen Messwert umgerechnet:

$$Messwert = \frac{Rohwert}{7680000} \cdot Endwert - Nullverschiebung$$

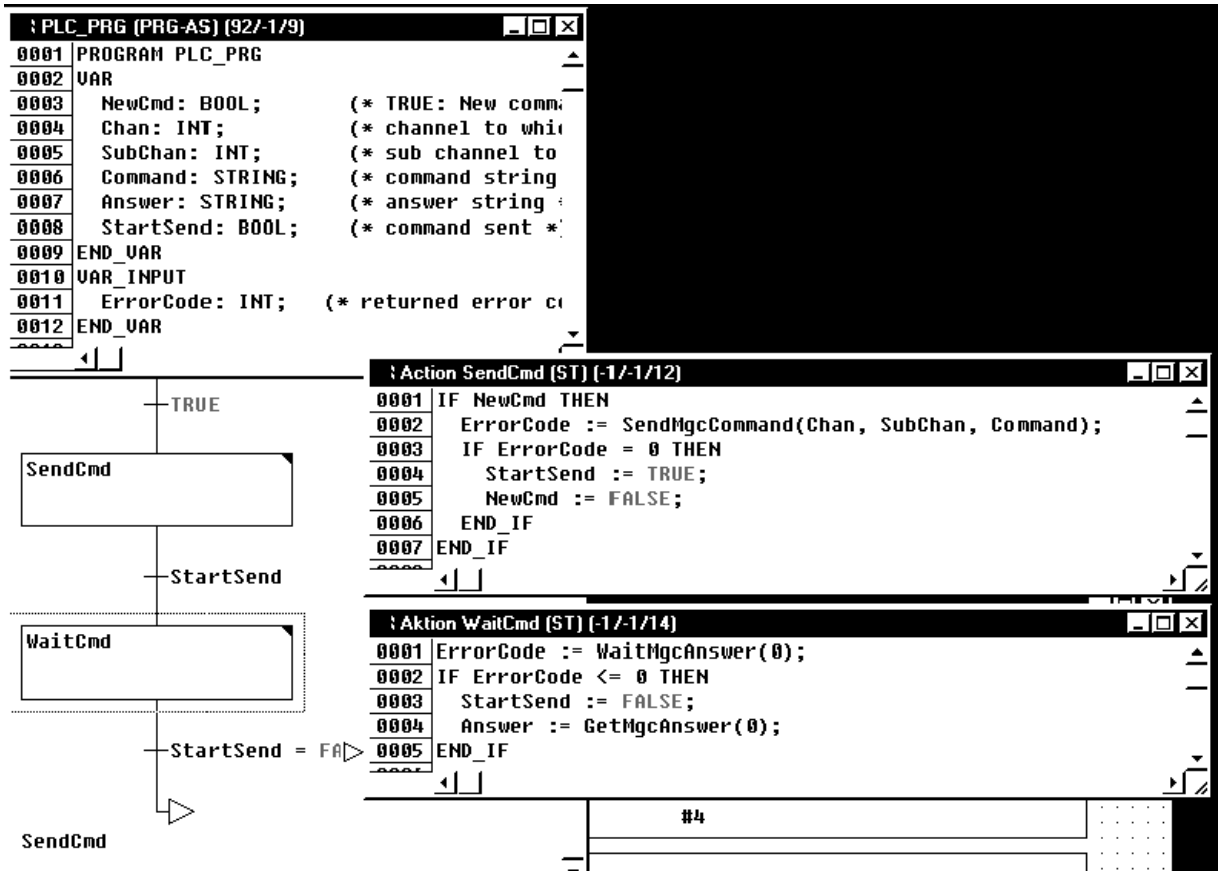
Endwert und Nullverschiebung sind in der Verstärkerskalierung festgelegt (siehe auch Kap.7.3) und können vom Verstärker mit der Funktion `GetChannelInfo()` abgefragt werden.

6.2 Befehle senden

Der ML70B kommuniziert mit anderen Verstärkereinschüben über die Funktionen `SendMgcCommand()`, `WaitMgcAnswer()` und `GetMgcAnswer()`.

Im ersten Schritt wird der Kommando−String durch Aufrufen der Funktion `SendMgcCommand()` an den gewünschten Kanal bzw. Unterkanal geschickt.

Anschließend wird die Funktion `WaitMgcAnswer()` so lange abgefragt, bis das Ergebnis 0 (OK) bzw. <0 (Fehler) vorliegt. Dann kann der Antwortstring mit der Funktion `GetMgcAnswer()` abgeholt werden.



7 Hardwarekomponenten konfigurieren

7.1 Analogausgänge

Die Analogausgänge des ML70B werden durch die Funktionen `SetAnalogOutputReal()` und `SetAnalogOutputInt()` angesprochen.

Parameter 1 entspricht der Nummer des Analogausganges (1 =V_{O1} auf BNC-Buchse oder AP75/AP78; 2=V_{O2} auf AP75/AP78).

Bei der Funktion `SetAnalogOutputReal()` können Sie als zweiten Parameter direkt die gewünschte Ausgangsspannung in Volt (−10....+10.) als REAL−Variable eingeben.

Sind aus Zeitgründen Berechnungen in REAL zu aufwendig, kann die Funktion `SetAnalogOutputInt()` verwendet werden. Als zweiter Parameter wird ein Wert zwischen −30000 ... 30000 als INT−Wert übergeben. −30000 entsprechen dabei −10 V, 30000 ensprechen 10 V.

7.2 Leuchtdioden der Frontplatte

Die LEDs auf der Frontplatte des ML70B lassen sich über die Funktion `SetLED()` ansprechen.
In Parameter 1 wird dabei die Nummer der LED (1...7) entsprechend der Nummerierung auf der Frontplatte übergeben. In Parameter 2 steht der gewünschte Zustand der LED: 0=Aus, 1=Rot, 2=Gelb

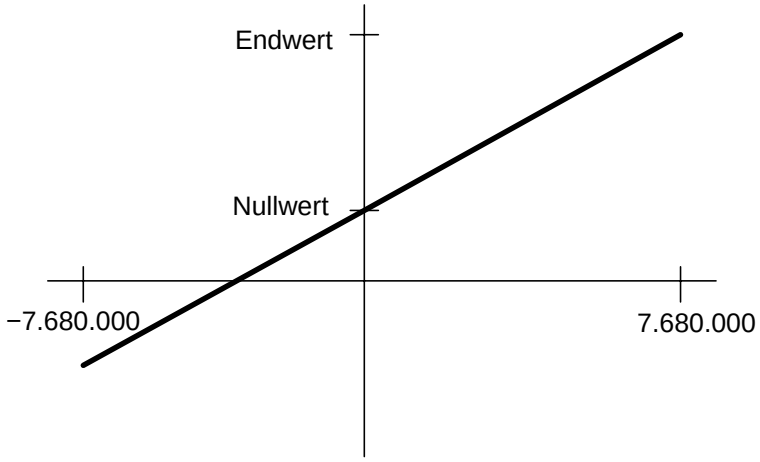
7.3 Berechnete Werte ausgeben

Die Funktion `OutputSignal()` gibt die berechneten Werten auf dem internen Datenbus aus. Als erster Parameter wird die Nummer des gewünschten Unterkanals übergeben, auf dem der ML70B das Signal ausgeben soll. Der zweite Parameter ist der auszugebende Wert als REAL-Wert.

HINWEIS

Um über catman® bzw. das AB22A eine korrekte Darstellung der Werte zu erhalten, muss vor der Ausgabe des ersten Wertes eine Skalierung angegeben werden.

Im MGCplus werden alle Messwerte als 24−Bit−Integerzahlen übermittelt. Der Zahlenbereich geht dabei von −7 680 000 ... +7 680 000. Jede REAL−Zahl wird nach folgendem Schema zur Übertragung in eine 24−Bit−Integer−Zahl gewandelt.



Auf der Empfängerseite wird die Integerzahl wieder in REAL−Werte gewandelt.

Für die Umrechnung müssen Endwert und Nullwert mit der Funktion `SetScaling()` dem ML70B angegeben werden:

$$Messwert = \frac{Linkwert}{7.680000} \bullet Endwert - Nullwert$$

8 Ansteuern der Anschlussplatten

8.1 AP71/CAN

Um die Bedienung der CAN−Schnittstelle zu verdeutlichen, sind die Beispielprogramme MGC_CanDemo, MGC_CanOpenDemo und MGC_SDOTerm beigefügt. Die CAN_Hardware wird über die Funktionen zur AP71 in der Bibliothek MGCplus.lib bedient. Um das Senden und Empfangen von CAN−Nachrichten zu vereinfachen, ist eine weitere Bibliothek MGCcan.lib beigefügt, die alle nötigen Funktionen beinhaltet.

8.1.1 Grundsätzliche Funktion des CAN-Treibers

Die beiden CAN−Schnittstellen der Anschlussplatte AP71 werden interruptgesteuert durch die Firmware bedient. Das IEC61131−3−Programm greift auf die CAN−Schnittstellen ausschließlich über die unten abgebildete Datenstruktur zu. CAN−Nachrichten werden über ein Ringpuffer gesendet und empfangen. Durch ein Vergleich von Lese− und Schreibindex kann das IEC−Programm feststellen, ob neue Nachrichten eingegangen sind.

```
0001 TYPE CAN_Interface :
0002 STRUCT
0003   pRx_Buffer    : POINTER TO CAN_Message;
0004   pTx_Buffer    : POINTER TO CAN_Message;
0005   pCallBacks    : POINTER TO CAN_CallBack;
0006
0007   CANMAX_RX_BUFFER: INT; (* Größe des Empfangsbuffers  n := (n + 1) MOD CANMAX_RX_BUFFER *)
0008   CANMAX_TX_BUFFER: INT; (* Größe des Sendebuffers  n := (n + 1) MOD CANMAX_TX_BUFFER *)
0009   CANMAX_CALLBACKS: INT; (* Größe des Callbackbuffers  n := (n + 1) MOD CANMAX_CALLBACKS *)
0010
0011   wBaudrate     : WORD; (* Beim Init-Aufruf einzustellende Baudrate *)
0012
0013   nRx_Index_DRV : INT := -1; (* Index des DRV's, für die zuletzt eingetragenen MSG in den Empfangsbuffer *)
0014   nTx_Index_DRV : INT := -1; (* Index des DRV's, für die als nächstes zu versendende  MSG aus dem Sendebuffer *)
0015   nWrite_DRV    : INT;       (* Index des DRV's, für die zuletzt eingetragenen MSG in den Empfangsbuffer
0016                               (für DRV's, die Eintragungen im Empfangsbuffer überschreiben können) *)
0017   nRx_Index_IEC : INT := -1; (* Index des IEC-Programms, für die als nächstes zu lesene MSG aus dem Empfangsbuffer *)
0018   nTx_Index_IEC : INT := -1; (* Index des IEC-Programms, für die zuletzt eingetragenen MSG in den Sendebuffer *)
0019   nWrite_IEC    : INT;  (* Index des IEC-Programms, für die zur Zeit beschriebene MSG in den Sendebuffer *)
0020   bOverWrite    : BOOL; (* hier kann der DRV anzeigen, daß ein Überlauf im Empfangsbuffer aufgetreten ist *)
0021   dwErrorCode   : DWORD; (* hier kann der DRV Errors eintragen, die dann im IEC-Programm ausgewertet werden können *)
0022   pArray        : ARRAY[0..32] OF WORD; (* not used *)
0023 END_STRUCT
0024 END_TYPE
0025
0026
```

Die Bibliothek MGCcan.lib enthält alle Funktionen für ein komfortables Auslesen der CAN−Nachrichten aus dem CAN_Interface.

Beispiel 1:

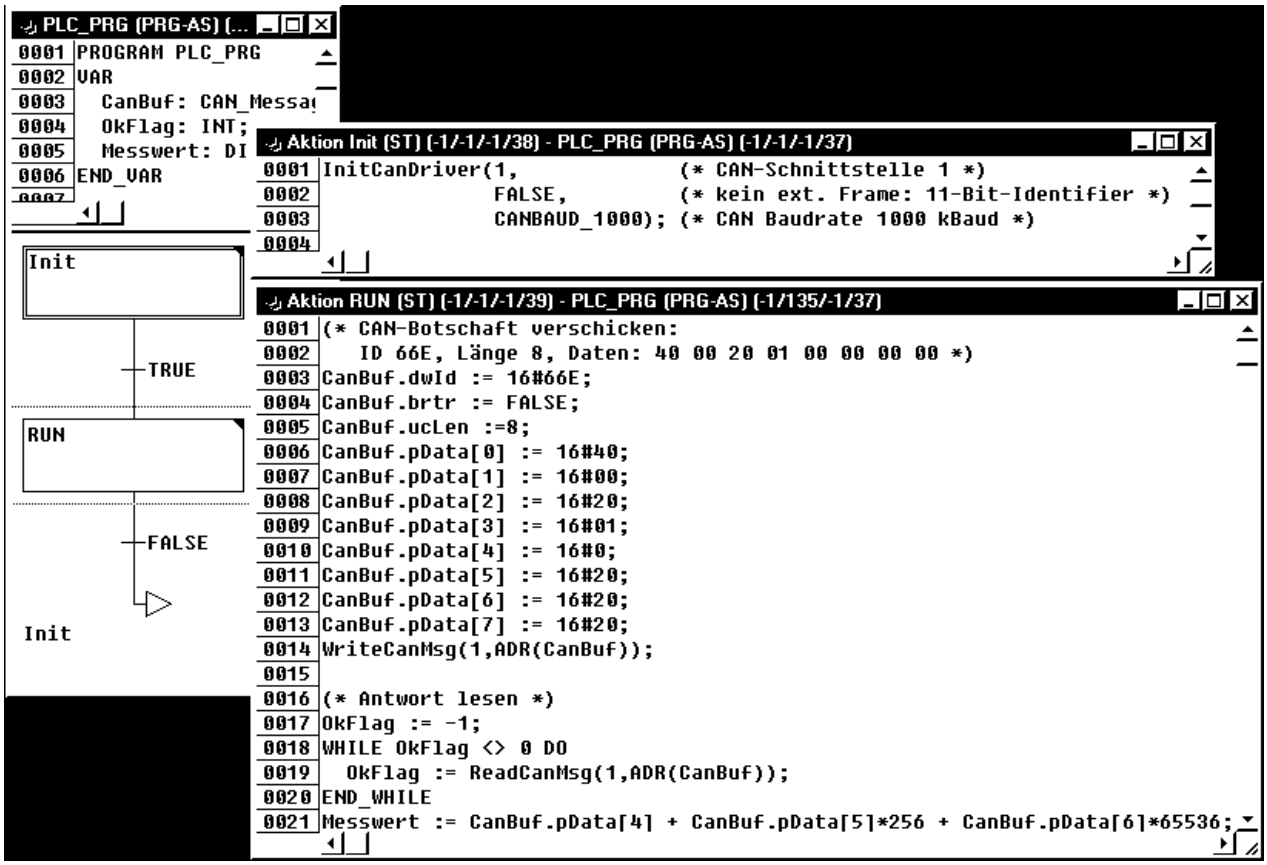
Kommunikation mit einem CAN-Knoten auf Low-Level-Ebene

Die Messwerte sollen kontinuierlich von einem CAN-Knoten eingelesen werden.

Im Schritt **Init** müssen Sie zunächst die gewünschte CAN-Schnittstelle initialisieren. In diesem Beispiel die Schnittstelle Nr. 1 auf der AP71. Mit der Funktion `InitCanDriver()` wird die Datenstruktur CAN_Interface sowie die Hardware für die CAN-Schnittstelle initialisiert.

Im Schritt **Run** wird die Datenstruktur CanBuf mit den gewünschten Werten gefüllt und mit der Funktion `WriteCanMsg()` wird die CAN−Botschaft anschließend verschickt.

Mit der Funktion `ReadCanMsg()` wird die Antwort eingelesen. Die Funktion gibt den Wert 0 zurück, wenn eine neue CAN−Nachricht eingetroffen ist.


PLC_PRG (PRG-AS) (...
0001 PROGRAM PLC_PRG
0002 VAR
0003 CanBuf: CAN_Messa
0004 OkFlag: INT;
0005 Messwert: DI
0006 END_VAR
Init
TRUE
RUN
FALSE
Init
Aktion Init (ST) (-1/-1/-1/38) - PLC_PRG (PRG-AS) (-1/-1/-1/37)
0001 InitCanDriver(1, (* CAN-Schnittstelle 1 *)
0002 FALSE, (* kein ext. Frame: 11-Bit-Identifier *)
0003 CANBAUD_1000); (* CAN Baudrate 1000 kBaud *)
0004
Aktion RUN (ST) (-1/-1/-1/39) - PLC_PRG (PRG-AS) (-1/135/-1/37)
0001 (* CAN-Botschaft verschicken:
0002 ID 66E, Länge 8, Daten: 40 00 20 01 00 00 00 00 *)
0003 CanBuf.dwId := 16#66E;
0004 CanBuf.brtr := FALSE;
0005 CanBuf.ucLen :=8;
0006 CanBuf.pData[0] := 16#40;
0007 CanBuf.pData[1] := 16#00;
0008 CanBuf.pData[2] := 16#20;
0009 CanBuf.pData[3] := 16#01;
0010 CanBuf.pData[4] := 16#0;
0011 CanBuf.pData[5] := 16#20;
0012 CanBuf.pData[6] := 16#20;
0013 CanBuf.pData[7] := 16#20;
0014 WriteCanMsg(1,ADR(CanBuf));
0015
0016 (* Antwort lesen *)
0017 OkFlag := -1;
0018 WHILE OkFlag <> 0 DO
0019 OkFlag := ReadCanMsg(1,ADR(CanBuf));
0020 END_WHILE
0021 Messwert := CanBuf.pData[4] + CanBuf.pData[5]*256 + CanBuf.pData[6]*65536;

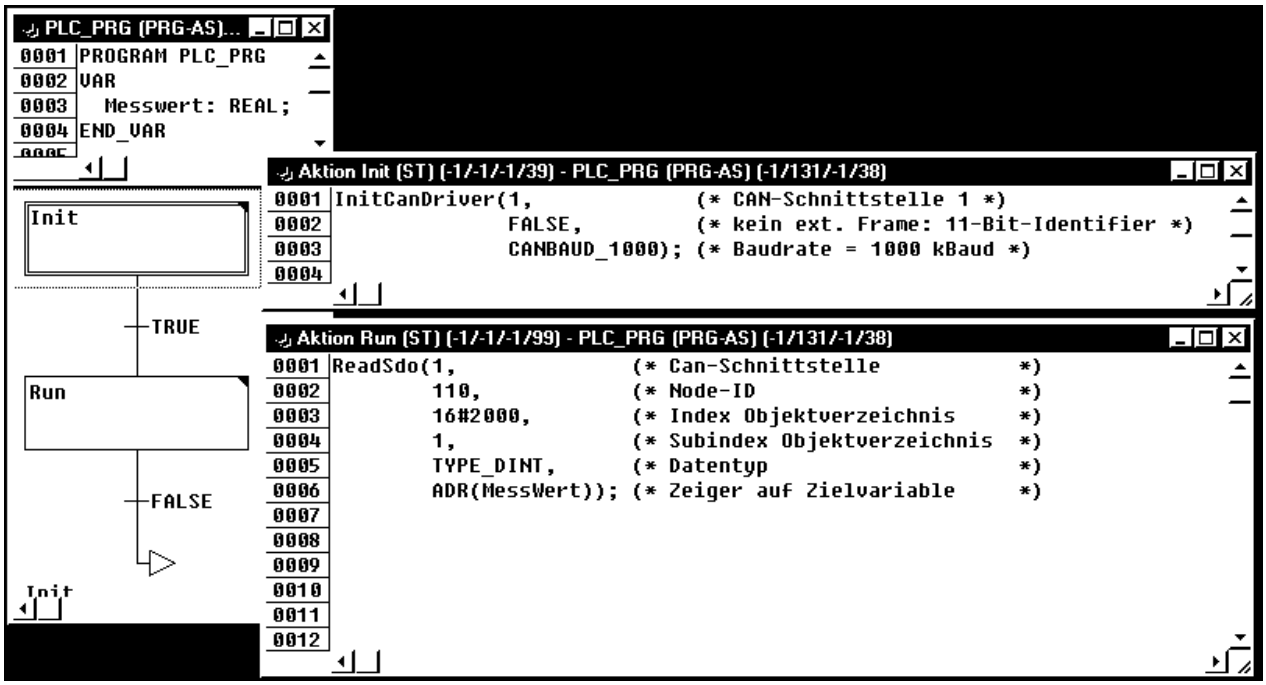
Beispiel 2:

Ansteuern eines CAN-Knotens über das CANopen-Protokoll

Ein Messwert soll über ein CANopen−SDO (Service Data Object) kontinuierlich einlesen werden.

Für das Lesen und Schreiben eines Objektes im Objektverzeichnis eines CANopen−Slaves befinden sich in der Bibliothek MGCcan die Funktionen ReadSdo() und WriteSdo().

Im unserem Beispiel hat der CANopen−Slave die Node−ID 110 und der Messwert liegt im Objektverzeichnis unter dem Index 16#2000 und Subindex 1 im Format DINT vor.



Im Schritt **Init** wird die CAN−Schnittstelle (einmalig) initialisiert . Im Schritt **Run** wird mit der Funktion `ReadSdo()` der Messwert des CANopen−Slaves eingelesen.

8.2 Serielle Kommunikation (AP72)

Mit dem ML70B können Sie bis zu vier serielle Schnittstellen ansteuern. Auf Steckplatz AP−A liegen die Schnittstellen Nr. 1 und 2, auf AP−B die Nr. 3 und 4. Sie können zwischen den drei Arten wählen:

- RS-232
- RS422 (Vollduplex 4−Draht−Verbindung mit Gegentakt−Signalen)
- RS485 (Halbduplex 2−Draht−Verbindung mit Gegentakt−Signalen).

Zum Senden und Empfangen von Daten müssen im IEC−Programm entsprechende Puffer bereitgestellt werden. Der eigentliche Sende− bzw. Empfangs-

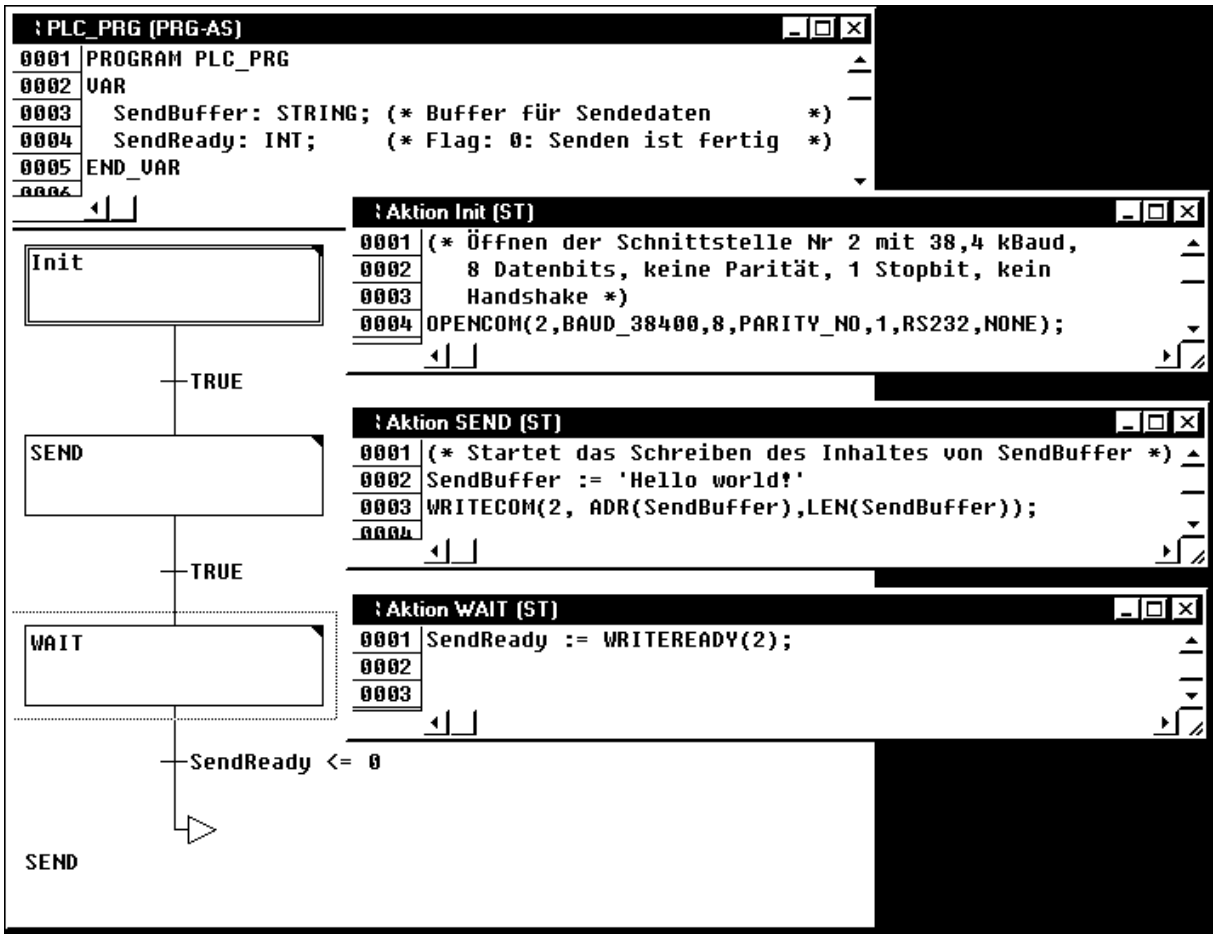
vorgang läuft (interruptgesteuert) im Hintergrund ab. Die Datentypen der Puffer sind abhängig von der jeweiligen Anwendung. Zweckmäßig sind meistens STRING oder ARRAY OF BYTE.

8.2.1 Daten senden

Zum Senden von Daten benötigen Sie die Funktionen `OpenCom()`, `WriteCom()` und `WriteReady()`.

Die Daten werden in drei Schritten gesendet:

1. Schnittstelle öffnen (siehe Bild; Zustand “Init”)
2. Daten bereitstellen und Sendevorgang starten (s.u. Zustand “SEND”)
3. Warten bis der Sendevorgang abgeschlossen ist (s.u. Zustand “WAIT”)



Das Programmbeispiel sendet endlos den String “Hello world!” auf der Schnittstelle 2. Die zur Einstellung der Schnittstelle nötigen Konstanten (Parameter der Funktion OPENCOM()) finden Sie in der Bibliotheksverwaltung im Register “Datentypen”.

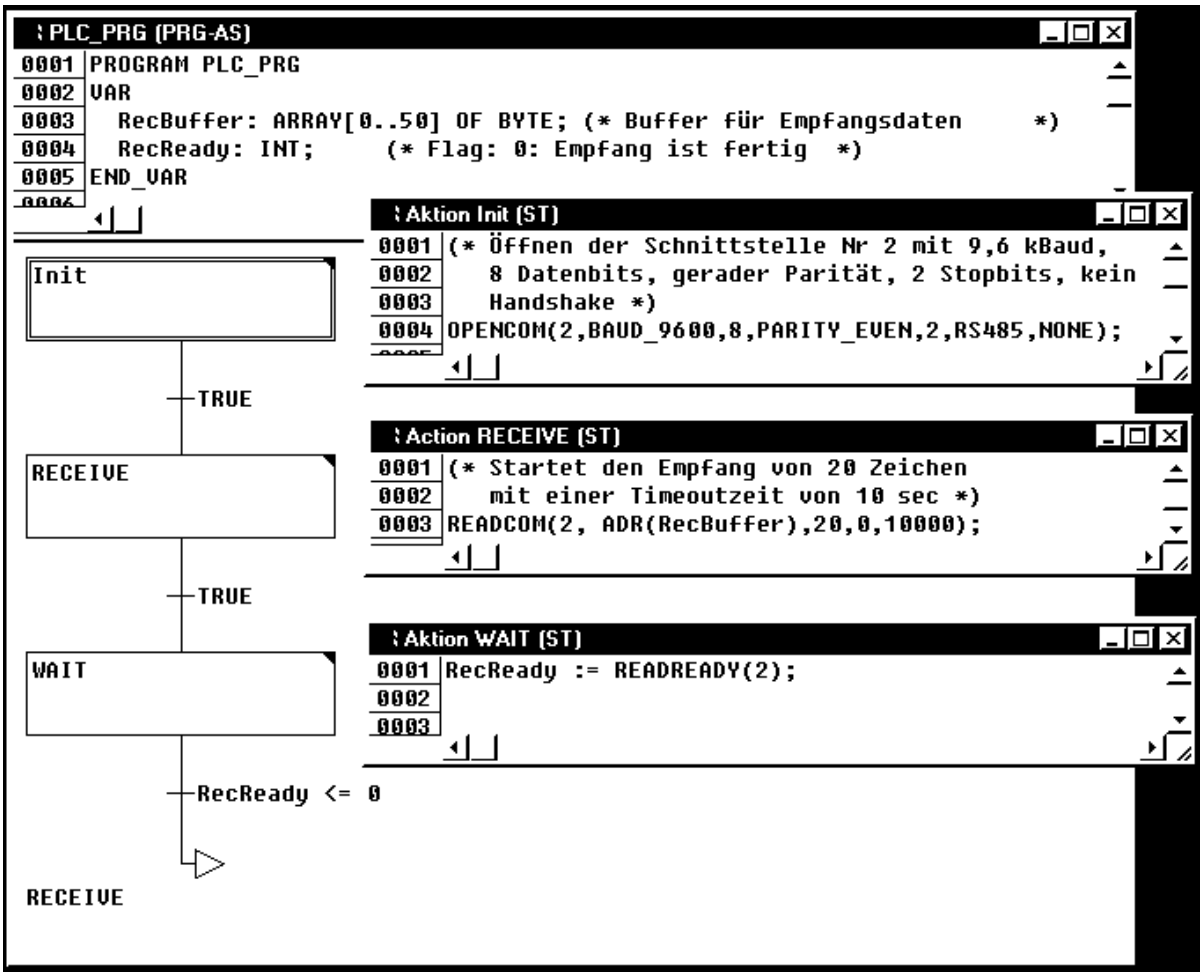
8.2.2 Daten empfangen

Zum Empfang von Daten benötigen Sie die Funktionen `OpenCom()`, `ReadCom()` und `ReadReady()`.

Die Daten werden in drei Schritten empfangen:

1. Schnittstelle öffnen (siehe Bild; Zustand “Init”)
2. Empfang starten (Zustand “RECEIVE”)
3. Warten, bis die erwarteten Daten empfangen worden sind (s.u. Zustand WAIT)

Beispiel 1:

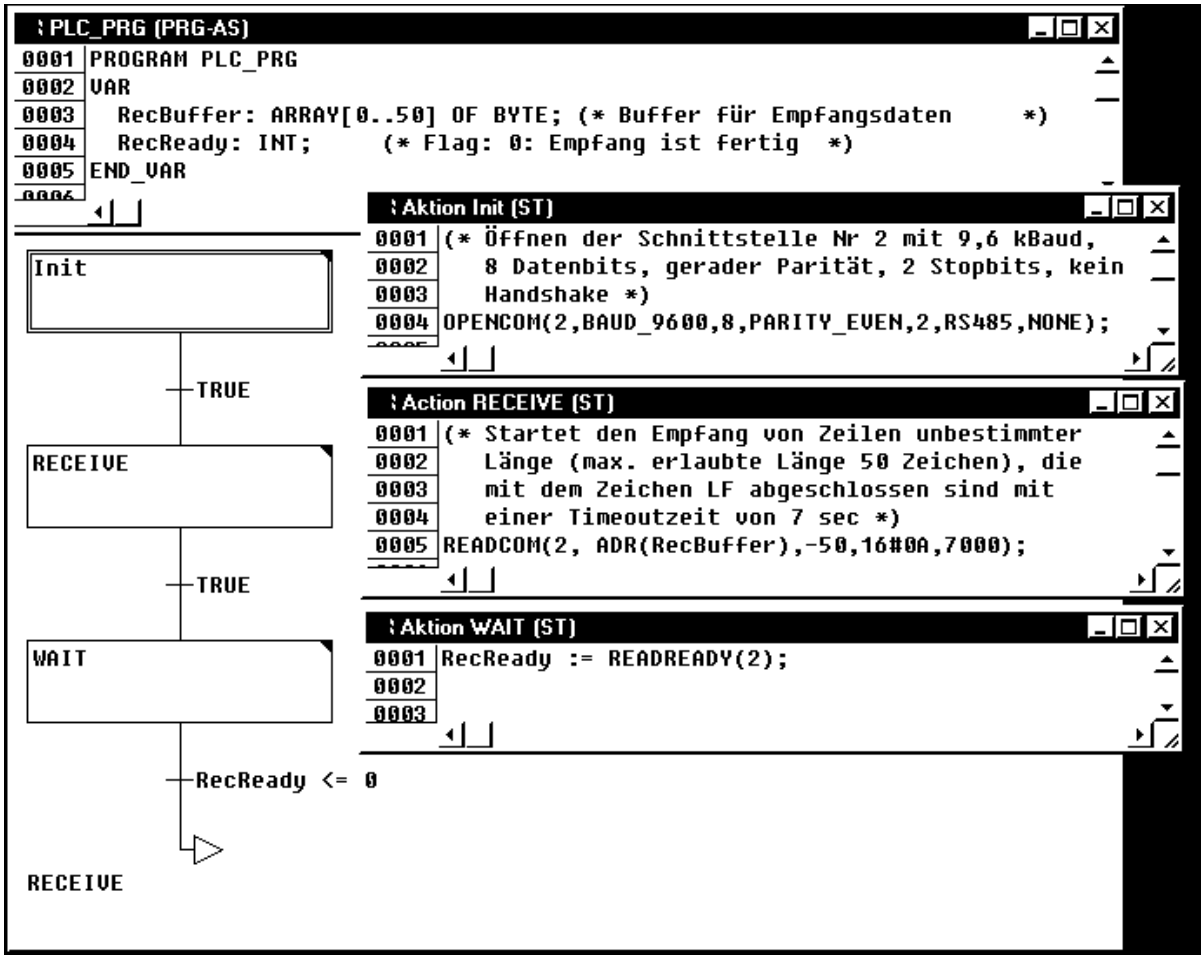


Das Programm empfängt endlos Blöcke von jeweils 20 Zeichen und schreibt diese in den Puffer “RecBuffer”.

Beispiel 2:

Falls die Länge des empfangenen Datenblocks unbekannt oder variabel ist, können Sie einen Empfangsmodus wählen, der den Empfang über eine Endekennung abbricht.

Als Datenlänge wird dabei eine negative Zahl übergeben. Der Betrag der Zahl ist dabei die maximal zur Verfügung stehende Pufferlänge. Weiterhin ist die gewünschte Endekennung zu übergeben.



8.3 Digitale Ein- und Ausgänge (AP75)

Mit der Funktion `SetOutputAP75()` (MGCplus.lib) können Sie die Digitalausgänge der AP75 (Achtung: Fremdspeisung erforderlich) setzen.

Der erste Parameter gibt die Nummer des Ausgangs an. Dabei sind die Ausgänge 1...8 auf der AP75 am Steckplatz A zu finden (Steckplatz direkt hinter dem ML70B) . Die Ausgänge 9..16 befinden sich auf der AP75 am Steckplatz B.

Der zweite Parameter gibt den Pegel an: TRUE = 24 V, FALSE = 0 V.

Mit der Funktion `GetInputAp75()` werden die Eingänge gelesen. Die Nummerierung ist analog zu den Digitalausgängen. Der Rückgabewert TRUE ent-

spricht einem Pegel > 10 V am Eingang, der Rückgabewert FALSE entspricht 0 V.

Beispiel:

Ausgang 3 der AP75 auf Steckplatz A soll auf HIGH−Pegel gesetzt werden:

SetOutputAp75(3,TRUE);

8.4 Analogausgänge AP78

Zum Ansteuern der Analogausgänge auf der AP78 dienen die Funktionen `SetAnalogOutputReal()` und `SetAnalogOutputInt()`.

Als Parameter 1 wird diesen Funktionen die Nummer des Analogausganges übergeben (3 ... 10; 1 und 2 sind für die Analogausgänge des ML70B reserviert).

Die Funktion `SetAnalogOutputReal()` hat als zweiten Parameter direkt die gewünschte Ausgangsspannung in Volt (−10....+10.) als REAL−Variable.

Wenn aus Zeitgründen Berechnungen in REAL zu aufwendig sind, können Sie die Funktion `SetAnalogOutputInt()` verwenden. Als zweiter Parameter wird ein Wert zwischen −30000 ... 30000 als INT−Wert übergeben. −30000 entsprechen dabei −10 V, 30000 ensprechen 10 V.

9 Dialoge erzeugen

9.1 Dialogstruktur

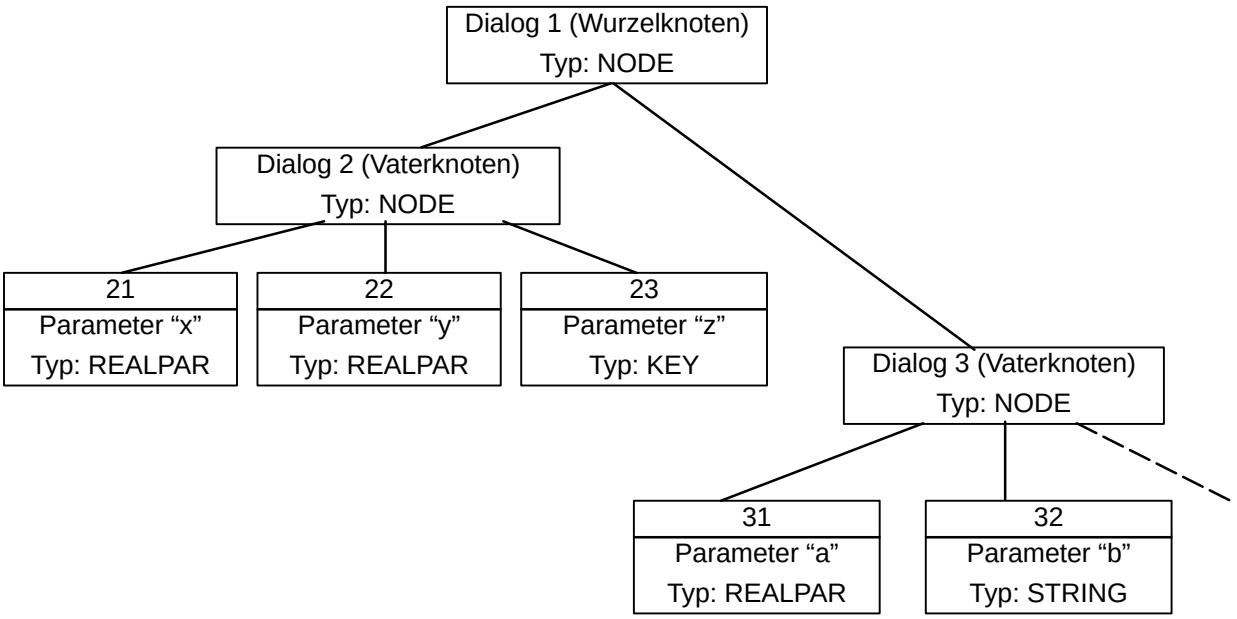
Der rechnende Einschub ML70B bietet Ihnen die Möglichkeit, Dialogfenster zu erstellen, die im MGCplus-Assistenten oder am Anzeige- und Bedienfeld ABxx aufgerufen werden können. Am ABxx können Sie dann im Einstellbetrieb gewünschte Dialoge den Funktionstasten F3 und F4 zuordnen. Ebenso ist es möglich Aktionen zu definieren, die im Messbetrieb über die Funktionstasten ausgelöst werden.

Die Dialoge werden aus verschiedenen Parametertypen gebildet. Parameter vom Typ “Knoten” bilden ein neues Dialogfenster, alle andere Parametertypen dessen Inhalt (Menüeinträge, Schaltflächen, Auswahlfelder, usw.).

Die Dialogparameter bilden eine Baumstruktur. An oberster Stelle ist der Wurzelknoten, welcher die Ausgangsanzeige bildet. Über weitere “Vaterknoten” kann jeweils eine Ebene tiefer in weitere “Unterdialoge” verzweigt werden. Die einem Knoten direkt untergeordneten Parameter bilden einen in sich geschlossenen Dialog.

Jeder Knoten und jeder Parameter werden durch eine eindeutige Nummer identifiziert. Der Wurzelknoten hat immer die Nummer 1. Sie können unterhalb des Wurzelknotens maximal 19 Dialoge erzeugen und dabei die Nummern 2 ... 999 (außer 20, diese Nummer ist bereits belegt) vergeben.

Für das Belegen der Funktionstasten des ABxx sind die Parameternummern 20 000 bis 20 499 reserviert. Der Wurzelknoten hat die Nummer 10 000. Die gewünschten Funktionen können Sie im Einstellbetrieb über die Einstellfenster “Anzeige”→ “F−Tasten” den Tasten F1 ... F4 zuordnen.



Um einen Dialog zu erzeugen, müssen Sie zunächst für jeden Parameter im IEC-Programm zwei Variablen deklarieren:

1. Die Variable selbst
2. Einen Funktionsblock mit den jeweiligen Eigenschaften und einer Funktion zum Eintrag in die Parameterliste

Parameter-typ	Datentyp für den Parame-ter	Datentyp für Parame-tereigenschaften	Bemerkungen
NODE	INT	CreateNode	Knoten
REALPAR	REAL	CreateRealPar	Parameter vom Typ REAL
INTPAR	INT	CreateIntPar	Parameter vom Typ INT
DINTPAR	DINT	CreateDintPar	Parameter vom Typ DINT
KEY	INT	CreateKeyPar	Schaltfläche (entspricht Windows-Button)
MENUE	INT	CreateMenuePar	Auswahlmenü (entspricht Windows-ComboBox)
TEXT	STRING	CreateTextPar	Parameter vom Typ STRING

Tab.9.1: Parametertypen

9.2 Parametereigenschaften

Jeder Parameter wird über mehrere Eigenschaften beschrieben und über einen Funktionsaufruf in die Parameterliste eingetragen. Jede Eigenschaft ist

mit einem Startwert vorbelegt. Es müssen also nur die Eigenschaften im Programm gesetzt werden, die von der Vorbelegung abweichen.

9.2.1 Eigenschaften des Parametertyps NODE

Knoten (NODEs) fassen Parameter zu Gruppen zusammen.

pValue: POINTER TO INT	Zeiger auf die Variable (die auch im IEC−Programm definiert sein muss), die den Zustand des Knotens enthält.
ParId: INT	Nummer des Knotens (1...19) Knoten 2..9 werden im SET−Betrieb des ABxx unter F3, Knoten 11..19 unter F4 dargestellt.
Name: STRING	Name des Knotens: Dieser Text wird im Dialog des ABxx dargestellt, wenn im SET−Betrieb die betreffende Taste (F3oder F4) gedrückt wird.
Root: INT	Hier wird eingetragen, zu welchem Wurzelknoten der Knoten gehört (für Knoten 1 muss hier nichts eingetragen werden)

Knoten erzeugen Sie mit der Funktion `CreateNode()`;

HINWEIS

Der Knoten mit der Nummer 1 (Wurzelknoten) ist im ML70B bereits angelegt.

9.2.2 Eigenschaften der Parametertypen DINTPAR, INTPAR, REALPAR

Mit diesen Parametertypen werden nummerische Parameter vom Typ DINT, INT , REAL dargestellt.

<Type> repräsentiert im Folgenden die jeweiligen Typen DINT, INT und REAL

pValue: POINTER TO <Type>	Zeiger auf die Variable (die auch im IEC−Programm definiert sein muss), die mit dem Parametrierdialog verändert werden soll
ParId: INT	Nummer des Parameters (2...999)
Name: STRING	Name des Parameters: Dieser Text wird im Dialog des ABxx zusammen mit dem Wert des Parameters dargestellt
Root: INT	Hier wird eingetragen, zu welchem Wurzelknoten der Parameter gehört.
MinVal: <Type>	Minimalwert, der bei der Einstellung des Parameters nicht unterschritten werden darf
MaxVal: <Type>	Maximalwert, der bei der Einstellung des Parameters nicht überschritten werden darf
EditWidth: INT	Editierweite
Decimals: INT	Anzahl der Nachkommastellen bei der Darstellung des Parameters
ScalFact: REAL	Skalierfaktor zur Darstellung am ABxx
Offset: REAL	Offset zur Darstellung im ABxx
Unit: STRING	Physikalische Einheit als String mit 4 Zeichen, die im Parametrierdialog zusammen mit dem Parameter angezeigt wird.
Flags: PARFLAGS	Spezielle Eigenschaften (bisher nicht unterstützt)

Über die Parameter ScalFact und Offset kann im Anzeiger der Wert des Parameters in einer anderen Skalierung dargestellt werden. Soll der Parameter unskaliert dargestellt werden (Normalfall), dann ist Scalfact = 1.0 und Offset = 0.0

Anzeigewert = Parameter * ScalFact − Offset

Die Parameter erzeugen Sie mit den Funktionsblöcken `CreateRealPar()`, `CreateIntPar()` und `CreateDintPar()`.

9.2.3 Eigenschaften des Parametertyps KEY

Mit dem Parametertyp KEY erzeugen Sie Schaltflächen in Dialogen. Mit Schaltflächen können Sie Aktionen auslösen oder weitere Unterdialoge öffnen.

Eine Schaltfläche erzeugen Sie mit dem Funktionsblock `CreateKey()`

pValue: POINTER TO INT	Zeiger auf die Variable (die auch im IEC−Programm definiert sein muss), die den Zustand der Schaltfläche im Dialog darstellt.
ParId: INT	Nummer des Parameters (2...999)
Name: STRING	Name der Schaltfläche: Mit diesem Text wird im Dialog des ABxx und des Assistenten die Schaltfläche beschriftet.
Root: INT	Nummer des Wurzelknotens, zu dem der Parameter gehört.
Flags: PARFLAGS	Spezielle Eigenschaften (bisher nicht unterstützt)

9.2.4 Eigenschaften des Parametertyps TEXT

Mit diesem Parametertyp erstellen Sie Texte in den Dialogen.

Einen Text erzeugen Sie mit dem Funktionsblock `CreateTextPar()`.

pValue: POINTER TO STRING	Zeiger auf den Text (der auch im IEC−Programm definiert sein muss), der mit dem Dialog verändert werden soll
ParId: INT	Nummer des Parameters (2...999)
Name: STRING	Name des Parameters: Dieser Text wird im Dialog des ABxx zusammen mit dem Wert des Parameters dargestellt
Root: INT	Nummer des Wurzelknotens, zu dem der Parameter gehört
Flags: PARFLAGS	spezielle Eigenschaften (bisher nicht unterstützt)

9.2.5 Eigenschaften des Parametertyps MENUE

Mit diesem Parametertyp erstellen Sie Auswahlfelder in Dialogfenstern. Ein Auswahlfeld wird durch eine INT−Variable dargestellt. Sie können bis zu 20 verschiedenen Werte auswählen, wobei jeder einzelne Wert durch einen Text repräsentiert wird.

Ein Auswahlfeld erzeugen Sie mit dem Funktionsblock `CreateMenuePar()`.

pValue: POINTER TO INT	Zeiger auf die Variable (die auch im IEC−Programm definiert sein muss), die mit dem Menü verändert wer den soll
ParId: INT	Nummer des Parameters (2...999)
Name: STRING	Name des Parameters: Dieser Text wird im Dialog des ABxx zusammen mit dem Wert des Parameters dargestellt
Root: INT	Nummer des Wurzelknotens, zu dem der Parameter gehört.
Flags: PARFLAGS	spezielle Eigenschaften (bisher nicht unterstützt)
Items: ARRAY[1..20] OF INT	Nummerische Werte, aus denen eine Auswahl getroffen wird
ItemTexts: ARRAY[1..20] OF STRING	Texte zu den nummerischen Werten

Beispiel 1:

Die Messwerte zweier Kanäle sollen multipliziert werden. Das Ergebnis soll auf dem Analogausgang ausgegeben und auf Unter− bzw. Überschreitung von Min− und Max.−Werten überprüft werden.

Eingestellt werden sollen:

P_{min}: untere Grenze

P_{max}: obere Grenze

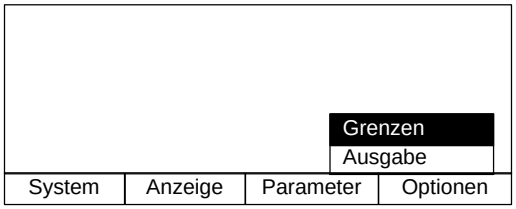
Löschen: Löschtaste

Rate: Ausgaberate des Analogausganges

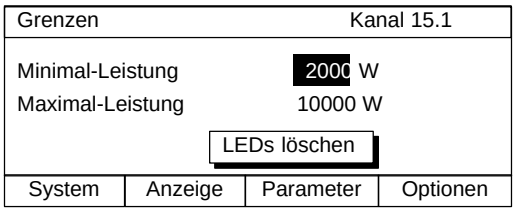
AnalogP1: Punkt 1 der Analogausgangskennlinie

AnalogP2: Punkt 2 der Analogausgangskennlinie

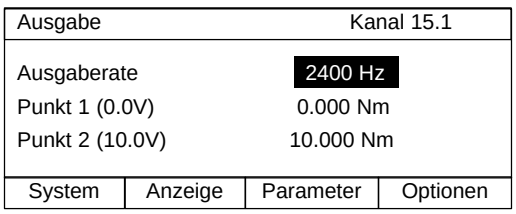
In der Anzeige sollen die Parameter in folgende Gruppen aufgeteilt werden:



Ausgangsdialog:
Auswahlfeld für “Grenzen” oder “Ausgabe”

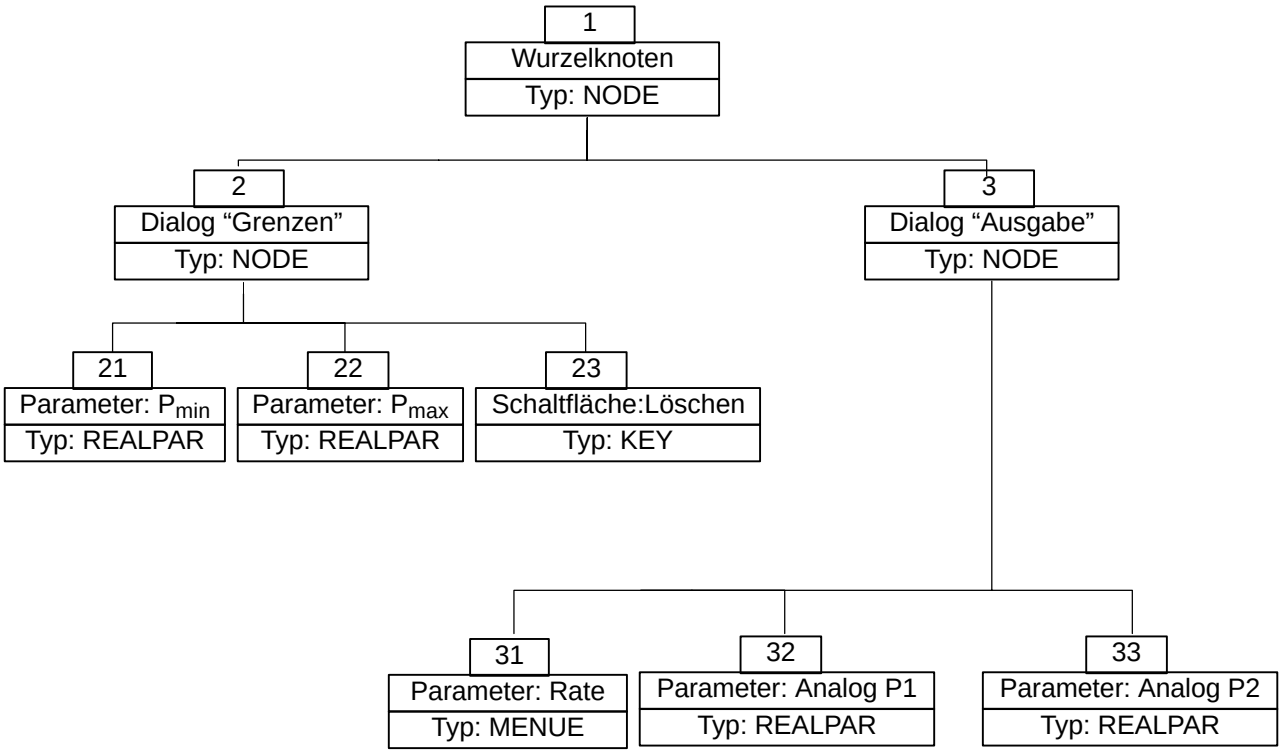


Im Fenster “Grenzen” werden die Parameter P_{min} und P_{max} und die Löschtaste dargestellt



Im Fenster “Ausgabe” werden die Ausgaberate sowie die Punkte 1 und 2 dargestellt

Die Dialoge bilden folgende Baumstruktur:



Um einen solchen Dialog zu definieren, müssen Sie die Eigenschaften der Parameter im IEC−61131−3−Programm codieren. In der Bibliothek MGCplus.LIB finden Sie in der Untergruppe “Parametering ML70B” Prototypen

von Funktionsbausteinen zur Beschreibung der unterschiedlichen Parametertypen.

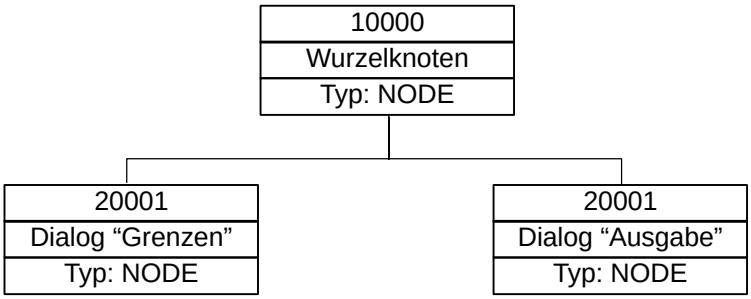
Um den dargestellten Parameterbaum im IEC−Programm zu codieren, ist folgender Quellcode notwendig. Dieser Code muss nur einmal durchlaufen werden. Wir empfehlen diesen Code Schritt INIT des Messprogrammes einzubinden (siehe Kapitel 4.5).

Erläuterungen	IEC61131−3 − Quelltext
Ein einzelner Dialog (Parametergruppe) besteht aus dem Knoten (hier Nummer 2 und 3) und den einzelnen Parametern. Jeder Parameter wird über die Eigenschaft “Root” mit dem Knoten verbunden	(* ------ Menü ”Grenzen” ------------ *) AttrNode2.pValue := ADR(Node2); (* Menüknoten ”Grenzen” *) AttrNode2.Name := ’Grenzen’; (* Anzeige-Text *) AttrNode2.ParId := 2; (* Parameter-Nummer *) AttrNode2.Root := 1; (* Wurzelknoten *) AttrNode2(); (* Erzeugen des Eintrags in der Param.-Liste *) AttrPmin.pValue := ADR(Pmin); (* Minimalmoment *) AttrPmin.ParId := 21; (* Parameter-Nummer *) AttrPmin.Name := ’Minimal-Leistung’;(* Anzeige-Text *) AttrPmin.Root := 2; (* Wurzelknoten *) AttrPmin.Decimals := 3; (* Anzahl Nachkommastellen für Anzeiger *) AttrPmin.Unit := ’Nm’; (* physikalische Einheit *) AttrPmin(); (* Erzeugen des Eintrags in der Param.-Liste *) AttrPmax.pValue := ADR(Pmax); (* Maximakraft *) AttrPmax.ParId := 22; (* Parameter-Nummer *) AttrPmax.Name := ’Maximal-Leistung’;(* Anzeige-Text *) AttrPmax.Root := 2; (* Wurzelknoten *) AttrPmax.Decimals := 3; (* Anzahl Nachkommastellen für Anzeiger *) AttrPmax.Unit := ’Nm’; (* physikalische Einheit *) AttrPmax(); (* Erzeugen des Eintrags in der Param.-Liste *) AttrKey1.pValue := ADR(ClearLeds); (* Taste ”LEDs löschen *) AttrKey1.ParId := 23; (* Parameter-Nummer *) AttrKey1.Name := ’LEDs löschen’; (* Anzeige-Text *) AttrKey1.Root := 2; (* Wurzelknoten *) AttrKey1(); (* Erzeugen des Eintrags in der Param.-Liste *) (* ------ Menü ”Ausgabe” ------------ *) AttrNode3.pValue := ADR(Node3); (* Menü ”Ausgabe” *) AttrNode3.Name := ’Ausgabe’; (* Anzeige-Text *) AttrNode3.ParId := 3; (* Parameter-Nummer *) AttrNode3.Root := 1; (* Wurzelknoten *) AttrNode3(); (* Erzeugen des Eintrags in der Param.-Liste *) AttrMenueRate.pValue := ADR(Rate); (* Auswahl-Menü ”Ausgaberate” *) AttrMenueRate.ParId := 31; (* Parameter-Nummer *) AttrMenueRate.Name := ’Ausgaberate’;(* Anzeige-Text *) AttrMenueRate.Root := 3; (* Wurzelknoten *) AttrMenueRate.Items[1] := 1; (* Zahlenwert 1. Menüeintrag *) AttrMenueRate.ItemTexts[1] := ’2400 Hz’;(* Text 1. Meüeintrag *) AttrMenueRate.Items[2] := 2; (* Zahlenwert 2. Menüeintrag *) AttrMenueRate.ItemTexts[2] := ’1200 Hz’;(* Text 2. Meüeintrag *) AttrMenueRate.Items[3] := 4; (* Zahlenwert 3. Menüeintrag *) AttrMenueRate.ItemTexts[3] := ’600 Hz’;(* Text 3. Meüeintrag *) AttrMenueRate.Items[4] := 8; (* Zahlenwert 4. Menüeintrag *) AttrMenueRate.ItemTexts[4] := ’300 Hz’;(* Text 4. Meüeintrag *) AttrMenueRate(); (* Erzeugen des Eintrags in der Param.-Liste *) AttrP1.pValue := ADR(AnalogP1); (* Kennlinienpunkt 1 Analogausgang *) AttrP1.ParId := 32; (* Parameter-Nummer *) AttrP1.Name := ’Punkt 1 (0.0 V)’; (* Anzeige-Text *) AttrP1.Root := 3; (* Wurzelknoten *) AttrP1.Decimals := 3; (* Anzahl Nachkommastellen für Anzeiger *) AttrP1.Unit := ’Nm’; (* physikalische Einheit *) AttrP1(); (* Erzeugen des Eintrags in der Param.-Liste *) AttrP2.pValue := ADR(AnalogP2); (* Kennlinienpunkt 2 Analogausgang *) AttrP2.ParId := 33; (* Parameter-Nummer *) AttrP2.Name := ’Punkt 2 (10.0 V)’; (* Anzeige-Text *) AttrP2.Root := 3; (* Wurzelknoten *) AttrP2.Decimals := 3; (* Anzahl Nachkommastellen für Anzeiger *) AttrP2.Unit := ’Nm’; (* physikalische Einheit *) AttrP2(); (* Erzeugen des Eintrags in der Param.-Liste *)

Beispiel 2:

Bei einem Anwenderprogramm sollen die Funktionstasten mit “Start” und “Stopp” belegt werden.

Dieses Beispiel ergibt folgende Baumstruktur:



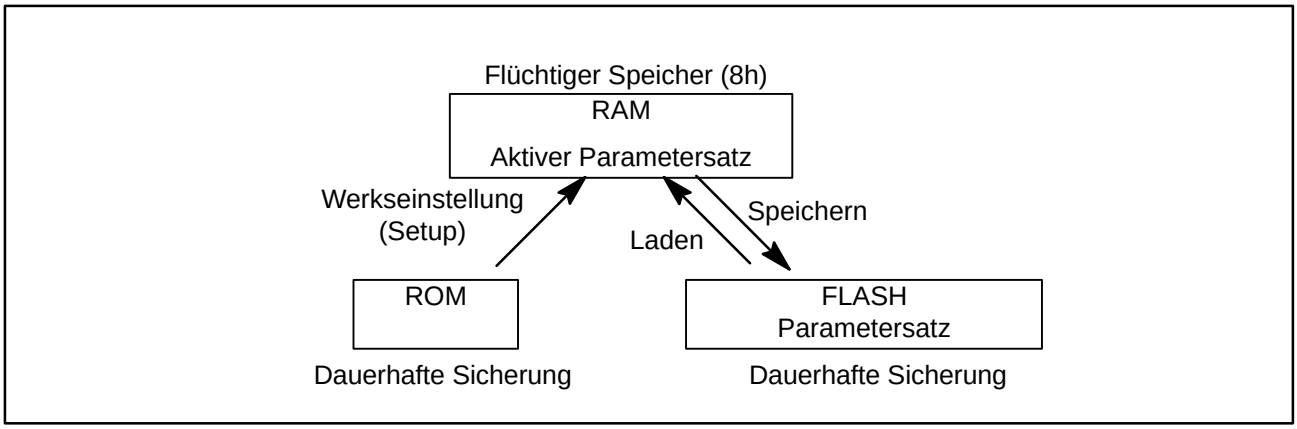
IEC−Quelltext:

```
(* −−− Funktionstasten −−−−−− *)
AttrNodeF.pValue := ADR(NodeF);      (* Knoten F−Tasten *)
AttrNodeF.Name := ’F−Tasten’;        (* Anzeige−Text *)
AttrNodeF.ParId := 10000;            (* Parameter−Nummer *)
AttrNodeF();                         (* Erzeugen des Eintrages in der Parameterliste *)
AttrKeyF1.pValue := ADR(FStart);     (* Taste ”Start” *)
AttrKeyF1.ParId := 20000;            (* Parameter−Nummer *)
AttrKeyF1.Name := ’Start’;           (* Anzeige−Text *)
AttrKeyF1.Root := 10000;             (* Wurzelknoten *)
AttrKeyF1();                         (* Erzeugen des Eintrages in der Parameterliste *)
AttrKeyF2.pValue := ADR(FStopp);     (* Taste ”Stopp” *)
AttrKeyF2.ParId := 20001;            (* Parameter−Nummer *)
AttrKeyF2.Name := ’Stopp’;           (* Anzeige−Text *)
AttrKeyF2.Root := 10000;             (* Wurzelknoten *)
AttrKeyF2();                         (* Erzeugen des Eintrages in der Parameterliste *)
```

Ausgelöst werden die Funktionen vom ABxx, indem die Werte der Variablen `Fstart` bzw. `Fstopp` bei jedem Tastendruck hochgezählt werden. Das IEC−Programm muss also zyklisch die Werte der Variablen abfragen, um die gewünschte Funktion ausführen zu können.

10 Einstellparameter speichern

Die im Kapitel 9 beschriebenen Dialogparameter können Sie speichern. Alle MGCplus-Einschübe speichern nach folgendem Prinzip:

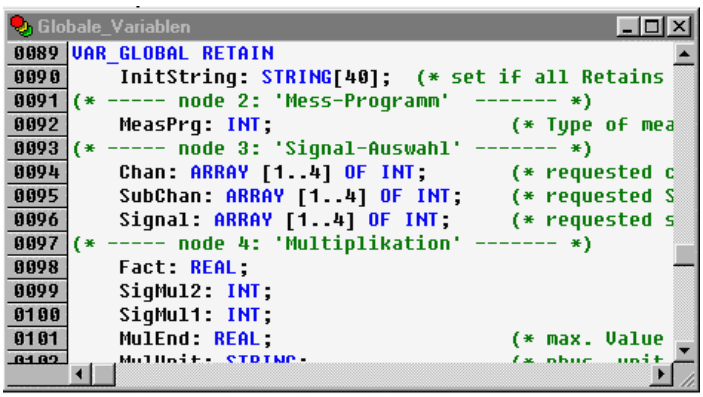


Die Einstellungen eine Einschubes werden immer aus dem im Arbeitsspeicher (RAM) liegenden Parametersatz gelesen, wo sie bis zu 8 Stunden stromausfallsicher gehalten werden können.

Beim Speichern des aktuellen Parametersatzes wird eine Kopie im nichtflüchtigen Speicher (Flash) abgelegt. Beim Laden wird der Parametersatz aus dem Flash in den Arbeitsspeicher kopiert. Beim Laden der im Werk voreingestellten Parameter werden diese aus dem ROM in den Arbeitsspeicher geladen.

Die Auswirkungen dieses Prinzips auf ihr Anwenderprogramm sollen hier mit einem Beispiel verdeutlicht werden. Als Beispiel dient die Standardapplikation im Verzeichnis `..\CodeSys..\Targets\HBM\ML70B_DemoPrj`.

Alle Variablen, die permanent gespeichert werden sollen, müssen im sogenannten “RETAIN−Bereich” als globale Variablen abgelegt werden. Dieser Bereich wird beim Aus− und Einschalten des MGCplus **nicht** initialisiert. Hier gespeicherte Daten sind für ca. 8 Stunden stromausfallsicher gespeichert.

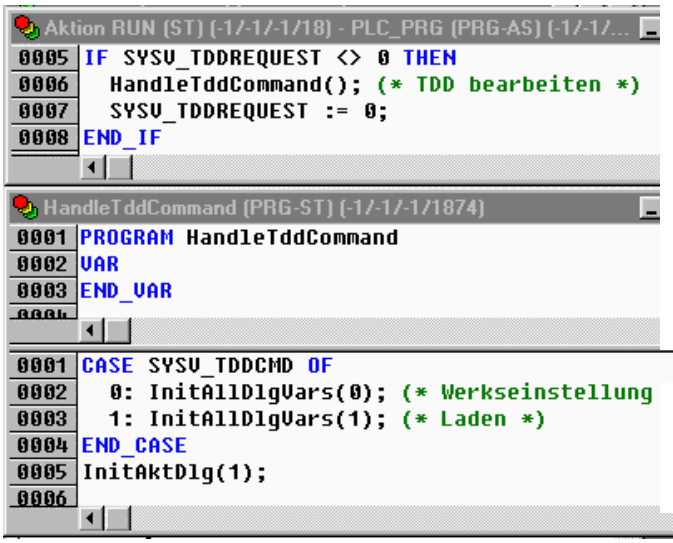

Globale_Variablen
0089 VAR_GLOBAL RETAIN
0090 InitString: STRING[40]; (* set if all Retains
0091 (* ----- node 2: 'Mess-Programm' ------- *)
0092 MeasPrg: INT; (* Type of mea
0093 (* ----- node 3: 'Signal-Auswahl' ------- *)
0094 Chan: ARRAY [1..4] OF INT; (* requested c
0095 SubChan: ARRAY [1..4] OF INT; (* requested S
0096 Signal: ARRAY [1..4] OF INT; (* requested s
0097 (* ----- node 4: 'Multiplikation' ------- *)
0098 Fact: REAL;
0099 SigMul2: INT;
0100 SigMul1: INT;
0101 MulEnd: REAL; (* max. Value

Wird vom Anzeige− und Bedienfeld oder einer externen Schnittstelle (z. B. Assistent) der Befehl **Speichern** an den ML70B geschickt, kopiert das Be-

triebssystem des ML70B den RETAIN-Bereich in den nicht flüchtigen Speicher. Wird der Befehl **Laden** an den ML70B geschickt, kopiert das Betriebssystem den Parametersatz aus dem FLASH in den RETAIN-Bereich.

Die Funktion **Werkseinstellung Laden** müssen Sie in ihrem Programm ausführen, da die Werkseinstellung des Anwenderprogramms dem Betriebssystem nicht bekannt sein kann.

Über die Systemvariablen. `SYSV_TDDREQUEST` und `SYSV_TDDCMD` kann das IEC−Anwenderprogramm abfragen, ob ein TDD−Befehl (Speichern/Laden) empfangen wurde.



Die Variable `SYSV_TDDREQUEST` wird vom Betriebssystem hochgezählt, wenn ein TDD−Kommando empfangen wurde.
Hier wird zyklisch die Systemvariable abgefragt. Bei Empfang eines TDD−Befehls wird das Programm `HandleTddCommand` aufgerufen.
In der Variable `SYSV_TDDCMD` steht die Art des empfangenen Befehls:
0: Werkseinstellung laden
1: Laden
2: Speichern

Hier wird auch der Befehl “Laden” vom Programm bedient, weil beim Wechsel einiger Parameter weitere programminterne Einstellungen vorgenommen werden müssen.

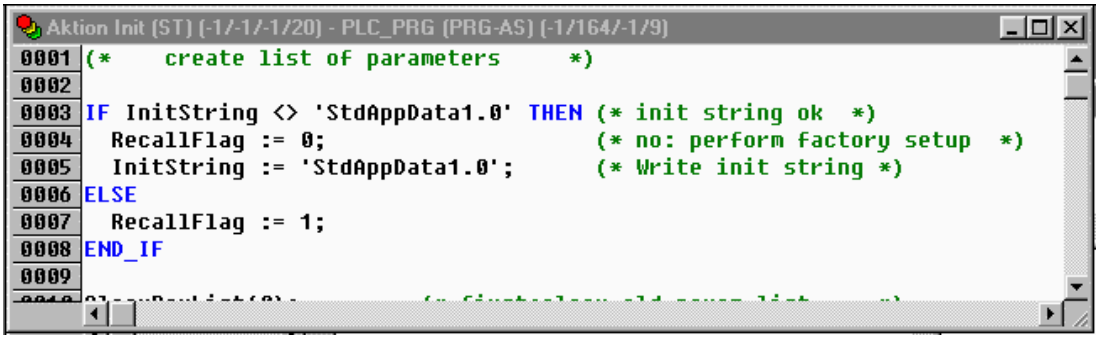
Beim Speichern des RETAIN-Bereiches wird vom Betriebssystem eine Prüfsumme berechnet.

Nach dem Einschalten des MGCplus überprüft das Betriebssystem zunächst, ob die Prüfsumme der RETAIN-Daten stimmt. Ist die Prüfsumme fehlerhaft, wird automatisch der Parametersatz aus dem FLASH ins RAM kopiert.

HINWEIS

Diese Programmgestaltung garantiert noch nicht, dass der Parametersatz auch zum momentan ausgeführten Programm passt. Der Parametersatz könnte z. B. von einem vorher geladenen Programm sein.

Deshalb sollte Ihr Programm zu Beginn folgende Prüfung durchführen:


Aktion Init (ST) (-1/-1/-1/20) - PLC_PRG (PRG-AS) (-1/164/-1/9)
0001 (* create list of parameters *)
0002
0003 IF InitString <> 'StdAppData1.0' THEN (* init string ok *)
0004 RecallFlag := 0; (* no: perform factory setup *)
0005 InitString := 'StdAppData1.0'; (* Write init string *)
0006 ELSE
0007 RecallFlag := 1;
0008 END_IF
0009

Wenn im String `InitString` nicht ein zur Applikation passender Wert steht, wird automatisch die Werkseinstellung ausgeführt, damit alle Variablen des Parametersatzes mit sinnvollen Werten voreingestellt sind.

11 Mehrprogrammbetrieb

Sie können mehrere voneinander unabhängige Programme scheinbar “gleichzeitig” abarbeiten (multitasking). Im Registerblatt “Ressourcen” befindet sich der Eintrag “Taskkonfiguration”. Hier können Sie verschiedene Programme verschiedenen Tasks zuordnen. Es sind bis zu 32 unterschiedliche Tasks möglich. Sie können jeder einzelnen Task eine Aufruffrequenz im ms−Raster und eine Priorität zuordnen.

Die genaue Beschreibung der Bedienung der Task−Konfiguration finden Sie in der Onlinehilfe von CoDeSys unter **Inhaltsverzeichnis −> Ressourcen −> Taskkonfiguration**.

HINWEIS

Die Aufruffrequenz des IEC-Programmes im ML70B beträgt 2400 Hz (siehe Kapitel 4.1).

Die Aufruffrequenz der Tasks müssen Sie in Millisekunden angeben. Eine äquidistante Messwertübertragung ergibt sich nur dann, wenn die folgende Formel für *Teiler* einen ganzzahligen Wert ergibt:

$$Teiler = \frac{2400\ Hz}{1000} \cdot Zeitintervall\_Taskaufruf[ms]$$

Die höchste Ausgabefrequenz für exakt äquidistante Messwertausgabe bei Multitasking beträgt damit 200 Hz.

12 Sonderfälle

12.1 Äquidistante Messwertausgabe

In manchen Anwendungen kann es nötig sein, Messwerte in zeitgleichen Abständen (äquidistant), auszugeben. Folgende Schritte sind nötig, um die maximale Rechenzeit des Programms zu ermitteln und diese Zeit anschließend als Ausgaberate festzulegen.

1. Loggen Sie sich ein.
2. Löschen Sie die Systemvariable SYSV_MAXEXECTIME im Fenster “Globale Variablen”.
3. Starten Sie das Programm.
4. Lesen Sie die globale Variable SYSV_MAXEXECTIME aus.
5. Fügen Sie am Anfang des IEC−Programms die Zeile SYSV_REQEXECTIME := <Wert> ein. Setzen Sie für <Wert> den ermittelten Wert ein.

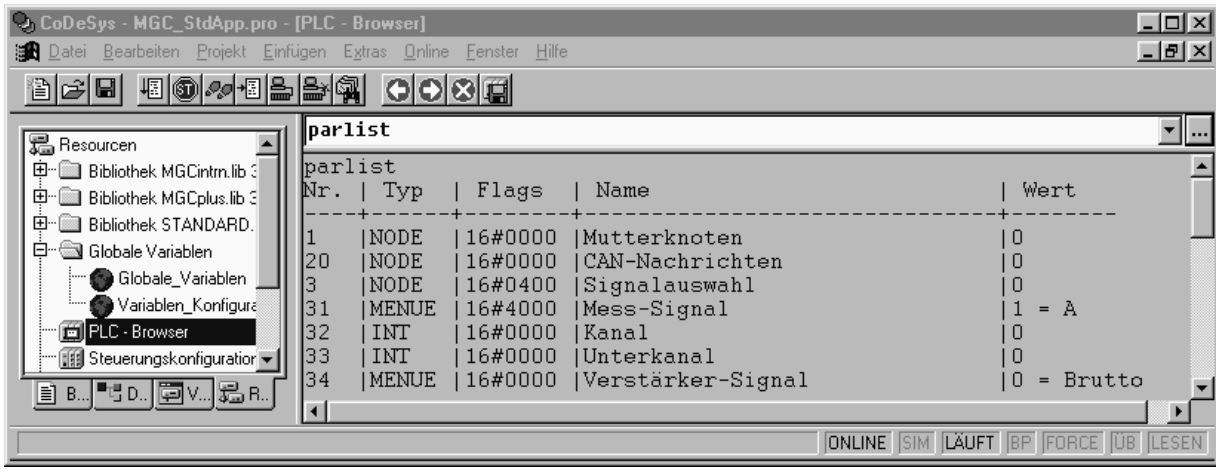
Damit ist die maximal aufgetretene Rechenzeit als Ausgaberate festgelegt.
Die Zeit wird dabei in Inkrementen von 1/2400 Hz = 416,6 µs angegeben.

12.2 Anzahl der Unterkanäle ändern

Die Anzahl der Unterkanäle des ML70B können Sie zwischen 1 und 128 Unterkanälen wählen.
Senden Sie über die externe Schnittstelle der Befehl **PAR9990, <Anzahl Unterkanäle>.** Schalten Sie anschließend das MGCplus aus− und wieder ein. Danach ist die Anzahl der Unterkanäle dauerhaft auf den gewünschten Wert eingestellt.

13 Debugfunktionen (PLC−Browser)

Das Programmiersystem CoDeSys hat einige Debug-Funktionen, die Fehlersuchen in Programmen erleichtern sollen. Im Registerblatt “Ressourcen” befindet sich der Eintrag PLC−Browser. Der Browser besteht aus einer Kommandozeile und einem Ausgabefenster. Die in der Kommandozeile eingegebenen Befehle liefern ihr Ergebnis im Ausgabefenster.


CoDeSys - MGC_StdApp.pro - [PLC - Browser]
Datei Bearbeiten Projekt Einfügen Extras Online Fenster Hilfe
Resourcen
Bibliothek MGCintrn.lib
Bibliothek MGCplus.lib
Bibliothek STANDARD.
Globale Variablen
Globale_Variablen
Variablen_Konfigura
PLC - Browser
Steuerungskonfiguration
parlist
parlist
Nr. | Typ | Flags | Name | Wert
1 |NODE |16#0000 |Mutterknoten |0
20 |NODE |16#0000 |CAN-Nachrichten |0
3 |NODE |16#0400 |Signalauswahl |0
31 |MENUE |16#4000 |Mess-Signal |1 = A
32 |INT |16#0000 |Kanal |0
33 |INT |16#0000 |Unterkanal |0
34 |MENUE |16#0000 |Verstärker-Signal |0 = Brutto
ONLINE SIM LÄUFT BP FORCE ÜB LESEN

Eine ausführliche Bedienungsanleitung des Browsers finden Sie in der Onlinehilfe von CoDeSys unter **Inhaltsverzeichnis −> Ressourcen −> PLC-Browser**. Weitere Eingabehilfe zu den Kommandos finden Sie über die mit “...” gekennzeichnete Schaltfläche rechts neben der Eingabezeile des PLC−Browsers.

Folgende Kommandos stehen im PLC-Browser zur Verfügung:

Kommando	**Beschreibung**
comp	Speichervergleich
compd	Speichervergleich mit Memory Dump
mem	Hexdump eines Speicherbereiches
memc	Hexdump relativ zur Startadresse des Codes auf der Steuerung
memd	Hexdump relativ zur Datenbasisadresse auf der Steuerung
memset	Speicherbereich setzen: <Start−Adresse>, <Fill−Byte>, <Länge>
metrics	PLC metrics anzeigen
reflect	Aktuelle Kommandozeile spiegeln, zu Testzwecken
dpt	Data−Pointer Tabelle lesen.
ppt	POU−Tabelle auslesen.
pid	Projekt−ID lesen.
pinf	Projekt−Info lesen
?	Liste der verfügbaren Kommandos ausgeben
cycletimes	Anzeige der Zykluszeiten des Programms auf dem ML70B
parlist	Anzeige der Parameterliste des ML70B
signals	Anzeige einer Liste aller vom ML70B angeforderten Signale

14 Systemvariablen

Variablenname	Typ	Bedeutung
SYSV_ActualExecTime	WORD	Ausführungszeit des letzten Schrittes gemessen in Zeitinkrementen von 1/2400 Hz = 416 µs
SYSV_MaxExecTime	WORD	Max. Ausführungszeit aller letzten Schritte gemessen in Zeitinkrementen von 1/2400 Hz = 416 µs
SYSV_ReqExecTime	WORD	Ausführungszeit des letzten Schrittes gemessen in Zeitinkrementen von 1/2400 Hz = 416 µs
SYSV_ParChangeFromML70B	WORD	Wird mit jeder Parameteränderung von außen (ext. Schnittstelle CP oder AB) vom ML70B inkrementiert
SYSV_ParChangeFromIEC	WORD	
SYSV_TddRequest	WORD	Wird beim Empfang eines TDD−Befehles von außen (ext. Schnittstelle CP oder AB) inkrementiert
SYSV_TddCmd	WORD	Letzter empfangener TDD−Befehl
SYSV_TddPar	WORD	Letzter mit TDD empfangener Parametersatz

15 Fehlermeldungen

Viele Funktionen aus der Bibliothek MGCPLUS.LIB geben Fehlercodes zurück, die einen genaueren Rückschluss auf die Fehlerursache geben können. Negative Werte sind Fehler, positive Werte sind Statusinformationen.

Fehlercode	Bedeutung
0	Kein Fehler
−100	Fehler beim Aufruf von RemovePar(): dieser Parameter existiert nicht
−101	Der angegebene Wurzelknoten existiert nicht
−102	Speicherüberlauf beim Anlegen eines Parameters
−103	Zu viele Items in einem Knoten
−104	Itemliste ist bereits leer
−105	Item kann nicht gelöscht werden (1 und 20)
−106	Wurzelknoten 20 nicht erlaubt!
−107	Knotennummern 1 und 20 sind reserviert
−108	falsche Parameternummer
200	ActivateSignals() läuft noch
−210	OpenComPort(): falsche PrtNr: diese serielle Schnittstelle existiert nicht
−211	OpenComPort(): Parity−Info falsch
−212	OpenComPort(): Anzahl Stopbits falsch
−213	OpenComPort(): Baudrate existiert nicht
−214	OpenComPort(): Anzahl Datenbits falsch
−215	Com−Port nicht geöffnet
−216	ReadCom() Parameter BUFF: ungültiger Pointer
−217	Timeout beim seriellen Empfang überschritten
−218	Zeilenlänge beim seriellen Empfang überschritten
−219	Empfangsfehler auf der seriellen Schnittstelle (Parity, Framing oder Overrun−Error) aufgetreten
−220	Hardware−Handshake nur bei RS232 möglich
−221	Software−Handshake bei RS485 nicht möglich
−230	Falsche Kanalnummer
−231	Falsche Unterkanalnummer
−233	SetScaling(): Endscale 0 ist nicht erlaubt
−234	SetScaling(): Es sind 0...5 Nachkommastellen erlaubt
240	Warte auf Antwort
−241	Es läuft bereits ein Befehlsprotokoll
−242	Timeout auf der internen MGC-Schnittstelle
−250	Zu viele Signale mit RequestSignal() angefordert
−251	ActivateSignals(): Signal nicht verfügbar
−260	SetAnalogOutput(): Analogausgang existiert nicht

16 Technische Daten ML70B

Analogausgänge		
Max. Anzahl Analogausgänge		2 (10 mit AP78)
Aktualisierungsrate Analogausgänge	Hz	2400
Nennspannung	V	± 10 V asymmetrisch
Zulässiger Lastwiderstand	kΩ	> 5
Innenwiderstand	Ω	< 5
Trägerrestspannung (76,6 kHz)	mVss	< 12
Langzeitdrift (über 48 h)	mV	< 3
Einfluss der Umgebungstemperatur bei 10°K: Messempfindlichkeit Nullpunkt	 % mV	 < 0,08 typ. 0,04 < 3 typ. 2
Programmierung		
Programmiersprache		IEC61131−3
Programmspeicher Daten (flüchtig)	kByte	224
Programmspeicher Daten (nicht flüchtig)	kByte	16
Programmspeicher Code (flüchtig) (2x vorhanden für Online−Change)	kByte	2 x 160
Programmspeicher Code (nicht flüchtig)	kByte	160
Speicher für Projekt−Sourcen (nicht flüchtig)	kByte	192
Aufruffrequenz des IEC−Programmes	Hz	2400, synchronisiert mit Messwertverarbeitung des MGCplus
Anzahl der Unterkanäle		1...128 (vom Benutzer einstellbar)
Nutzbare Rechenleistung		30.000 Float−Operationen/s oder 150.000 Integer−Operationen/s
Unterstützte Anschlussplatten		
Anzahl der ansteuerbaren Anschlussplatten		0,1 oder 2
Unterstützte Anschlussplattentypen		AP71 (2 CAN−Schnittstellen) AP72 (2 serielle Schnittstellen) AP75 (8 Digital−In, 8 Digital−Out, 24 V Pegel) AP78 (8 Analog Ausgänge)
Mechanik		
Nenntemperaturbereich	°C	−20 ... +60
Gebrauchstemperaturbereich	°C	−20 ... +60
Lagerungstemperaturbereich	°C	−25 ... +70
Betriebsspannungen	V	+14,6 ... +17,0 (< 90 mA) −14,6 ... –17,0 (< 100 mA) −7 ... −9 (<10 mA)
Kartenformat	mm	Europa 100 x 160
Breite	mm	20,3 (4 TE)
Anschlussstecker		indirekt DIN 41612

17 Technische Daten Anschlussplatten

17.1 AP71

CAN−Schnittstelle		
Anzahl der CAN−Schnittstellen		2
Baudrate	Baud	10 k, 20 k, 50 k, 125 k, 250 k, 500 k, 1 M
Leitungslänge	m	1000, 1000, 1000, 500, 250, 100, 25
Hardware−Busankopplung je CAN−Schnittstelle einzeln umschaltbar		ISO 11898−24 V “Low−Speed”
Anschlusstechnik		2x 9−pol. DSub, einzeln potentialgetrennt von Versorgung und Meßmasse
Mechanik		
Nenntemperaturbereich	°C	−20 ... +60
Gebrauchstemperaturbereich	°C	−20 ... +60
Lagertemperaturbereich	°C	−25 ... +70
Betriebsspannungen	V	+14,6 ... +17,0 (<130 mA) −14,6 ... –17,0 (<140 mA) −7 ... −9 (<10 mA)
Kartenformat	mm	Europa 100 x 160
Breite	mm	20,3 (4 TE)
Anschlussstecker		Indirekt DIN 41612

17.2 AP72

Schnittstellen		
Baudrate	kBaud	9,6; 19,2; 38,4; 57,6; 115,2
Galvanische Trennung	V	Typ. 500
Anschlusstechnik		9−pol.Sub−D Buchse
Mechanik		
Nenntemperaturbereich	°C	−20 ... +60
Gebrauchstemperaturbereich	°C	−20 ... +60
Lagertemperaturbereich	°C	−25 ... +70
Betriebsspannungen	V	+5 ... (< 100 mA)
Kartenformat	mm	102 x 112
Breite	mm	20,3 (4 TE)

17.3 AP75, AP78

ML78 + Anschlussplatte		AP78	AP75
Analog-Ausgänge			
Max. Anzahl Analog-Ausgänge		10 (2 Ausgänge filterbar, 1 davon auf Frontplatte ML78 zusätzlich zugänglich)	2 (beide Ausgänge filterbar 1 davon auf Frontplatte ML78 zusätzlich zugänglich)
Galvanische Trennung	V	Typ. 500[1)]	–
Auflösung D/A−Wandlung	bit	16	
Massesysteme		2[2)]	1, getrennt von dig. Massesystemen
Nennspannung	V	±10 asymmetrisch	
Zulässiger Lastwiderstand	kΩ	≥ 5	
Innenwiderstand	Ω	< 5	
Trägerrestspannung (76,6 kHz)	mV_{ss}	< 12	
Langzeitdrift (über 48 h)	mV	< 3	
Einfluss einer Umgebungstemperaturänderung von 10 K	% mV	< 0,08; typ. 0,04 < 3 typ. 2	
Digital-Eingänge			
Max. Anzahl Digital-Eingänge			8 (16)[3)]
Eingangsspannungsbereich	V		0 ... 24
Galvanische Trennung	V		Typ. 500
Low−Potential	V		< 5
High−Potential	V		>10
Massesysteme			1, getrennt von digitalem Ausgang
Digital-Ausgänge			
Max. Anzahl Digital-Ausgänge			8 (16)[3)]
Ausgangsspannungsbereich	V		0 ... 24
Ausgangsstrom	A		0,5
Kurzschlussstrom	A		1,5
Galvanische Trennung	V		Typ. 500
Reaktionszeit	ms		< 4
Massesysteme			1, getrennt von digitalen Eingängen
Speisung	V		24 (extern)
Mechanik			
Nenntemperaturbereich	°C	−20 ... +60	
Gebrauchstemperaturbereich	°C	−20 ... +60	
Lagertemperaturbereich	°C	−25 ... + 70	
Betriebsspannungen	V	+14,6 ... +17,0 (< 120 mA)/ −17,0 ... −14,6 (< 120 mA)/ −9,0 ... −7,0 (< 10 mA)	
Kartenformat / Breite	mm	Europa 160 x 100 / 20,3 (4TE)	
Anschlusstechnik		25 polig Sub−D	Steckbare Schraubklemmen

1) Die digital filterbaren Ausgänge sind nicht galvanisch getrennt!

2) 1 Massesystem für 2 digital filterbare Analogausgänge sowie 1 Massesystem für die restlichen 8 Analogausgänge

3) Bei Verwendung von 2 Anschlussplatten AP75: 16 Digital-Eingänge und 16 Digital-Ausgänge

4) Auf beiden Anschlussplatten sind jeweils beide Analogausgänge VO1 und VO2 verfügbar

Änderungen vorbehalten.
Alle Angaben beschreiben unsere Produkte in allgemeiner Form.
Sie stellen keine Eigenschaftszusicherung im Sinne des §459,
Abs. 2, BGB dar und begründen keine Haftung.

Hottinger Baldwin Messtechnik GmbH

Postfach 10 01 51, D-64201 Darmstadt
Im Tiefen See 45, D-64293 Darmstadt
Tel.: 061 51/ 8 03-0; Fax: 061 51/ 8039100
E−mail: support@hbm.com www.hbm.com

A0862-3.0 de

measurement with confidence